A Survey on the Application of Genetic Programming to Classification

Pedro G. Espejo, Sebastián Ventura, and Francisco Herrera

***Abstract*—Classification is one of the most researched questions in machine learning and data mining. A wide range of real problems have been stated as classification problems, for example credit scoring, bankruptcy prediction, medical diagnosis, pattern recognition, text categorization, software quality assessment, and many more. The use of evolutionary algorithms for training classifiers has been studied in the past few decades. Genetic programming (GP) is a flexible and powerful evolutionary technique with some features that can be very valuable and suitable for the evolution of classifiers. This paper surveys existing literature about the application of genetic programming to classification, to show the different ways in which this evolutionary algorithm can help in the construction of accurate and reliable classifiers.**

***Index Terms*—Classification, decision trees, ensemble classifiers, feature construction, feature selection, genetic programming (GP), rule-based systems.**

I. Introduction

TWO main approaches for learning are distinguished in machine learning: supervised and unsupervised learning. In supervised learning, attributes of data instances are divided into two types: inputs or independent variables and outputs or dependent variables. The goal of the learning process consists in predicting the value of the outputs from the value of the inputs. In order to accomplish this goal, a training set of data (data instances including the values of booth input and output variables with known values) is employed to guide the learning process. Regression and classification are two types of supervised learning tasks. In regression, there is a continuous-valued output to be predicted while in classification the output is discrete. In unsupervised learning, there is no distinction of type between the variables of the data instances. As a consequence, we cannot talk about training data since we cannot have a set of data with a known output. The goal of unsupervised learning is to find the intrinsic structure, relations, or affinities present in data. Examples of unsupervised learning tasks are clustering and association discovery. In clustering, the goal is to divide data into different groups, finding groups of data that are very different from each other, and whose members are very similar to each other. Association discovery consists in finding data values that appear together frequently.

Classification is one of the most studied problems in machine learning and data mining [1], [2]. It consists in predicting the value of a categorical attribute (the class) based on the values of other attributes (predicting attributes). A search algorithm is used to induce a classifier from a set of correctly classified data instances called the training set. Another set of correctly classified data instances, known as the testing set, is used to measure the quality of the classifier obtained. Different kinds of models, such as decision trees or rules, can be used to represent classifiers.

Genetic programming (GP) [3] is an evolutionary learning technique that offers a great potential for classification. GP is a very flexible heuristic technique that allows us to use complex pattern representations such as trees, for example, any kind of operation or function can be used inside that representation and domain knowledge can be used in the learning process, for example by means of a grammar.

GP and other evolutionary techniques has been successfully applied to different supervised learning tasks like regression [4] and unsupervised learning tasks like clustering [5]–[9] and association discovery [10]. However, we focus our review on the application of GP to classification. The distinctive features of GP make it a very convenient technique with regard to classification. The application of GP to classification offers some interesting advantages, the main one being its flexibility, which allows the technique to be adapted to the needs of each particular problem. GP can be employed to construct classifiers using different kinds of representations, e.g., decision trees, classification rules, discriminant functions, and many more. GP can be useful not only for inducing classifiers, but also for other preprocessing and postprocessing tasks aimed at the enhancement of classifiers. In fact, GP usually performs an implicit process of feature selection and extraction. Interpretability can be easily favored by the use of GP, since it can employ more interpretable representation formalisms, like rules. The automatic feature selection performed by GP and different mechanisms available for controlling the size of the resulting classifiers also contribute to improve interpretability. This paper reviews the works published in the literature, where GP is applied in some form to address classification tasks.

The rest of the paper is organized as follows. Section II briefly presents the GP and classification fields. Section III gives a preliminary overview of the possible ways of applying GP for classification. Section IV describes studies involving classification in which GP is used for feature selection and construction. Section V reviews the different approaches reported in the

Manuscript received July 1, 2008; revised February 10, 2009. First published November 24, 2009; current version published February 18, 2010. This work was supported by the Spanish Department of Research of the Ministry of Science and Technology (Projects TIN2008-06681-C06-01 and TIN2008-06681-C06-03). This paper was recommended by Associate Editor J. Lazansky.

P. G. Espejo and S. Ventura are with the Department of Computer Science and Numerical Analysis, University of Cordoba, Cordoba 14071, Spain (e-mail: pgonzalez@uco.es; sventura@uco.es).

F. Herrera is with the Department of Computer Science and Artificial Intelligence, University of Granada, Granada 18071, Spain (e-mail: herrera@decsai.ugr.es).

Digital Object Identifier 10.1109/TSMCC.2009.2033566

literature to obtain classifiers by means of GP. Section VI looks at the application of GP in the construction of ensemble classifiers. The drawbacks and advantages of using GP for classification are analyzed in Section VII, together with some guidelines on how to apply GP to classification tasks. Section VIII presents some concluding remarks and suggests topics for further research.

II. Background

In this section, we give a brief and general description of the two subjects that are the core of our review: GP and classification.

A. GP

The evolutionary algorithm (EA) paradigm is based on the use of probabilistic search algorithms inspired by certain points in the Darwinian theory of evolution [11]. Several different techniques are grouped under the generic denomination of EA. The essential features shared by all EAs are as given in the following.

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

GP is essentially considered to be a variant of genetic algorithms (GA) [12] that uses a complex representation language to codify individuals. The most commonly used representation schema is based on trees, although other options exist [3]. The original goal of GP, as its name implies, was the evolution of computer programs. However, nowadays GP is used to evolve other abstractions of knowledge, like mathematical expressions or rule-based systems, for example. GP individuals are usually seen as parse trees, where leaves correspond to terminal symbols (variables and constants) and internal nodes correspond to nonterminals (operators and functions). The set of all the nonterminal symbols allowed is called the function set, whereas the terminal symbols allowed constitute the terminal set. Two conditions must be satisfied to ensure that GP can be successfully applied to a specific problem: sufficiency and closure. Sufficiency states that the terminals and nonterminals (in combination) must be capable of representing a solution to the problem. Closure requires that each function of the nonterminal set should be able to handle all values it might receive as input. In practice, we often need to evolve programs that handle values of different types, and this makes it difficult to meet the closure requirement.

B. Classification

The goal in classification is to take an input vector x and to assign it to one of K discrete classes C_k where $k = 1, \ldots, K$ [13]. Each data instance is defined according to a set of attributes or variables. In order to solve a classification problem, a classifier needs to be obtained. The classifier is a model encoding a set of criteria that allows a data instance to be assigned to a particular class depending on the value of certain variables. A classification algorithm is a method for constructing a classifier. Supervised learning is an approach frequently used to obtain a classifier. It consists in using a set of data instances that are labeled with their correct classes in order to induce a classifier capable of classifying these data instances and, hopefully, other new ones. A great variety of algorithms have been proposed for learning classifiers from a set of training data. In many cases, this task is approached as a search process. The number of algorithms described in the literature used to carry out this task can be overwhelming. Three basic components are used to make a systematic categorization of the existing algorithms for classification that are based on a search approach [14].

1) The representation formalism. A certain piece of knowledge can be represented in different ways. The most common forms of representation for a classifier are decision trees and rules, but other choices have been reported in this paper, as we will see in Section V.
2) The preference criterion. Some kind of rule is needed in order to allow the system to make a choice for a particular model or set of parameters when several alternatives are available. In classification, the preference criterion is usually given by the accuracy rate, that is, the ratio of correctly classified examples. However, other factors can be considered, e.g., novelty, utility, or interpretability.
3) The search algorithm. The induction of a classifier is considered as a search process in a space of candidate classifiers. Once the representation formalism and the preference criterion have been specified, the task of obtaining a classifier becomes an optimization problem, where the search algorithm is employed to find the parameters optimizing the preference criterion.

Since GP is a search and optimization algorithm, it can be easily employed as the search algorithm for inducing a classifier. The great expressive capability of GP allows it to adapt to different representation formalisms. This way, individuals in the population can represent decision trees, classification rules, artificial neural networks (ANNs), and many more. Similarly, any preference criterion can be expressed in terms of the fitness function that guides the search process of GP. Because of the flexibility of GP, a very diverse range of approaches can be derived from this basic setting, as we will see in Section V.

While model extraction, that is, the construction of a classifier from a dataset, is the main task in the classification domain, there are some other related tasks that usually have to be addressed as well.

1) Data preprocessing. Model extraction techniques are not usually applied to the database in its original form. There are a great variety of preprocessing techniques available in order to prepare the data to take advantage of its maximum potential. Some of the issues to consider are the removal of noise or outliers, strategies for handling missing data, feature selection and construction, instance selection, data rebalancing and data projection, and normalization, to name a few.
2) Model interpretation and enhancement. Sometimes the classifier obtained is not readily usable, and some kind of postprocessing is convenient. Some of the questions to be addressed at this point can be the elimination of redundant knowledge, application of visualization techniques, evaluation of the model obtained, sorting of classification patterns according to some measure of interest, translation to a more interpretable form, combination of models, and any other way of improving the model obtained.

GP can be employed not only to induce classifiers, but also for preprocessing and model enhancement, as we will see in Sections IV and VI, respectively.

III. GP for Classification

Flexibility is one of the main advantages of GP, and this feature allows GP to be applied for classification in many different ways. As we have seen in Section II-A, individuals are usually represented as trees, and trees are one of the most general data structures, therefore, they can be tailored to fit the representation requirements in a wide range of problems and domains. But GP is not limited to tree-like individuals, because it also allows the use of other representations, like linear or graph structures. In addition, the nature of GP individuals, which include terminals (variables and constants) and nonterminals (operators and functions), gives them the ability not only to represent knowledge but also to perform computations, so that GP can be used in almost any classification-related task, not only in the core task of classifier induction, but also for preprocessing and postprocessing purposes.

At the preprocessing stage, data can be transformed in order to increase the quality of the knowledge obtained, and GP can be used to perform this transformation. The data transformation can consist in a selection of the attributes relevant to the classification problem at hand; a weighting of the attributes, in order to give each attribute a credit proportional to its importance in the final prediction; or it can consist in the construction of new predicting attributes by combining some of the original ones.

GP also offers a wide range of possibilities in the classifier induction task. As we have seen in Section II-B, GP can be readily fit to the main components of a model extraction algorithm: GP is a search algorithm, which can be employed to search in the space of classifiers to find the best one; the fitness function of GP is used as the preference criterion that drives the search process; and the flexibility of the GP representation allows it to employ many different kinds of models, with decision trees, classification rules, and discriminant functions being the most common choices. A decision tree [15] is composed of internal and leaf nodes. Since GP individuals are commonly encoded in a tree-like fashion, the application of GP to the evolution of decision trees is obvious. Each individual in the population can represent a decision tree classifier. A rule has two parts, the antecedent and the consequent. The antecedent contains a combination of conditions for the predicting attributes, and the consequent contains the predicted class. Usually, a condition is composed of a binary relational operator ($=$, $\neq$, $>$, $<$, $\geq$, $\leq$) comparing the value of an attribute with a constant or another attribute, although more complex conditions can easily be incorporated since GP individuals can employ any kind of operator. A discriminant function is a mathematical expression in which different kinds of operators and functions are applied to the attributes of a data instance that must be classified. The representation of a mathematical expression as a tree is again evident.

TABLE I
Confusion Matrix

		Predicted:	
		Positive	Negative
Real:	Positive	True Positives (TP)	False Negatives (FN)
	Negative	False Positives (FP)	True Negatives (TN)

No matter what representation formalism is used, the quality of classifiers must be measured by means of the fitness function. Usually, quality is based on accuracy, and it is often measured as the ratio between the number of correctly classified examples and the total number of examples, but other possibilities exist. Several widely used metrics for measuring accuracy, like precision, support, confidence, recall, sensitivity, specificity, and others, are based on the confusion matrix (see Table I). The true positives (TP) are positive instances that are classified as positive, the false negatives (FN) are positive instances classified as negative, the false positives (FP) are negative instances classified as positive, and the true negatives (TN) are negative instances classified as negative. Furthermore, other performance criteria can be taken in consideration in the fitness function, like novelty, interestingness, utility, or interpretability.

Occam's razor is a reasoning which is applied very often not only in classification, but virtually in every area of computer science. This principle basically states that when choosing between models with the same accuracy, less complex models should be preferred. This preference is due to several reasons, like enhanced interpretability and better generalization ability.

Classification performance can be enhanced employing several classifiers instead of just one. This is the basic idea of ensemble classifiers. Ideally, different base classifiers in an ensemble capture different patterns or aspects of a pattern embedded in the whole range of data, and then through ensembling, these different patterns or aspects are incorporated into a final prediction. Since this technique consists in the combination of the classification output of several classifiers, it can be regarded as a form of postprocessing of the models extracted. Two main issues, which are how to generate diverse base classifiers and how to combine base classifiers have to be addressed, and GP can be applied to both issues. The basic approach for using GP to generate diverse base classifiers consists in dividing the dataset

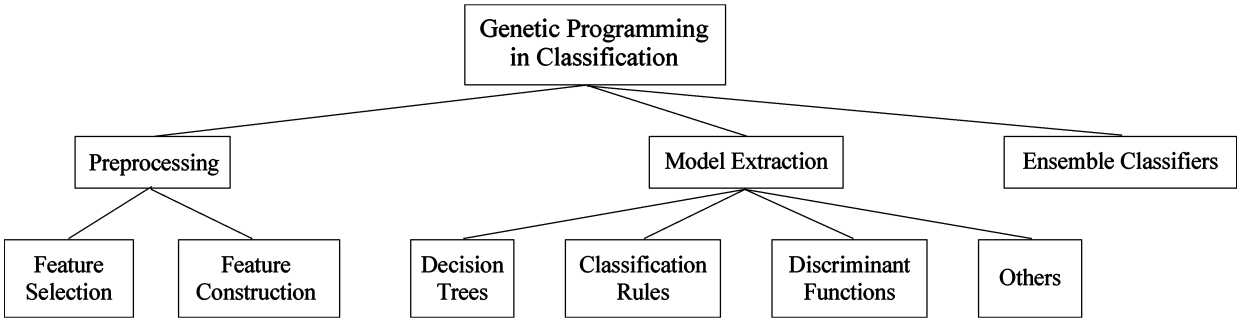

Fig. 1. Applications of GP in classification tasks.

into several subsets, where each of the subsets is used in an independent run of GP to construct each of the base classifiers. The application of GP to the combination of base classifiers is very similar to the construction of new features: the outputs of each of the base classifiers are at the leaf nodes of a tree GP individual, and these predictions are combined by means of the operations encoded in the nonterminal nodes of the tree.

Fig. 1 shows the classification tasks where GP can be applied. All these possibilities will be covered in our review.

In many problems, there are several goals which have to be optimized simultaneously. These goals are often conflicting, so that the optimization of one of the performance measures implies an unacceptably poor performance for other measures. For example, when feature selection is performed, we want to minimize the number of attributes employed and maximize the classification rate, but higher accuracy can usually be achieved when more features are employed [16]. A similar situation arises with instance selection, where a subset of the training set is selected in order to speed up the learning process, without compromising classification rates [17]. Accuracy and interpretability of classifiers are also conflicting goals; generally, the most accurate classifiers are the most complex and difficult for human beings to understand [18]. In these situations, a tradeoff solution must be sought, combining suboptimal but acceptable values for all performance measures. In this kind of problem, known as a multiobjective optimization (MO) problems, there is usually not a single solution, but instead a set of equivalent nondominated solutions, known as a Pareto front, composed of all the solutions where it is not possible to enhance some objectives without degrading some others. EAs can be applied to MO problems easily and suitably, since different individuals in the population can search for different solutions in parallel. When EAs are applied to this kind of problem, the term employed is multiobjective optimization evolutionary algorithm (MOEA) (MOGP if the EA is GP) [19], [20].

IV. GP for Feature Selection and Construction

In this section, we review the published works that apply GP to data preprocessing. All the studies included in this section make use of GP in order to carry out a particular kind of preprocessing known as transformation of representation, which, given the original vector of features F_0 and the training set L, consists in creating a representation F derived from F_0 that maximizes some criterion and is at least as good as F_0 with respect to that criterion [21]. The approaches that follow this scheme can be roughly divided into three categories.

1) Feature selection methods. Here, the resulting representation is a subset of the original one, i.e., $F \subseteq F_0$.
2) Feature weighting methods. In this case, the transformation method assigns weights to particular attributes. The weight reflects the relative importance of an attribute and may be utilized in the process of inductive learning.
3) Feature construction methods. Here, new features are created in some form, for instance as functional expressions that use the values of original features.

Furthermore, two important considerations must be taken into account. First, we must realize that the application of GP for inducing classifiers usually implies a feature selection process which is inherent to the evolution of classifiers. In GP, variable-length individuals are evolved. These individuals are usually tree-like structures where internal nodes are operators and functions, and leaf nodes are constants and variables, and these variables correspond to the features of the dataset. Since individuals have variable lengths, not all features must appear within an individual; in fact, it is more likely that only some of the features will be present in each individual. Therefore, implicit feature selection is performed as part of the evolutionary process. Second, in a way similar to the implicit feature selection outlined above, there is some kind of implicit feature construction behavior inherent to most GP-based classifiers. When the individual codification employed allows features from the dataset to be combined by means of the application of arithmetic operations, there is in some sense an implicit construction of features. Taking all these considerations into account, we have to make clear that this section is restricted to studies that use GP specifically and explicitly for preprocessing purposes.

Here, we provide some additional information about the references reviewed in this section. This information is focused on practical issues. On some occasions, the proposed GP-based system is applied to some specific kind of real problem. On other occasions, the proposed GP system is not aimed at any particular kind of problem, but its performance is tested with respect to a set of benchmark or synthetic datasets. This information is summarized in Table II. Similar tables are provided in Sections V and VI. The goal of these tables is to help researchers to find the references that are more relevant to their work.

TABLE II
APPLICATIONS OF GP-BASED DATA PREPROCESSING

Application area	References
Biology	[22], [23]
Medicine	[24], [25]
Finance	[26]
Law	[27]
Engineering	[28]
Object recognition	[29]
Benchmark	[30]–[35]
Synthetic	[26], [35]

A. Feature Construction

A basic idea is shared by all the studies included in this section: the new constructed features are codified by tree-like individuals, where arithmetic operators and functions appear in the internal nodes, and the original features and maybe constants correspond to leaf nodes.

Conceptually, preprocessing is a step previous to the model extraction task. However, in practice, two alternative approaches can be followed in the implementation of a preprocessing method.

1) The filter approach. Here, the preprocessing is performed before proceeding with the model induction algorithm. Usually, some kind of statistical, logical or information content criterion is used as the basis for the preprocessing.
2) The wrapper approach. Preprocessing is carried out while the model induction algorithm is being applied, that is, the execution of the preprocessing and the classification algorithms are interleaved. The quality of the model induced by the classification algorithm is used to bias the preprocessing.

1) Filter Approach: In [30], four fitness functions designed to choose the best constructed features are proposed, and four different classification techniques are subsequently used to build the classifier. The four fitness measures are information gain, the gini index, a combination of information gain and the gini index and chi-square [1]. The four classification algorithms which are later applied are three decision tree algorithms (C5, CART, and CHAID) and a multilayer perceptron. In this proposal, each individual in the population is a tree representing a new constructed feature. When the evolutionary process ends, a single new feature is obtained, which is added to the original data, and this augmented dataset is used to induce a classifier applying some of the chosen classification algorithms.

In [24] and [28], the proposed fitness function is designed to measure the scattering between classes, that is, the degree of difference between them. The classifiers to be applied later are a neural network and a support vector machine (SVM). Each individual in the population is a tree representing a new constructed feature.

A fitness function based on information entropy is employed in [31], where a class-wise orthogonal transformation is applied to the original features previous to GP-feature extraction, producing a set of transformed features which is added to the original features in order to make up a variable terminal pool to be used by the GP algorithm as a basis for constructing new features. Each individual in the population is a tree representing a new constructed feature as a combination of the variables from the pool. The system constructs as many new features as there are classes in the dataset, and these are the only features employed for the final classification.

2) Wrapper Approach: In [23], a GP feature constructor is wrapped around the k-nearest neighbors (kNN) algorithm. Each individual is a forest composed by n trees, n being the number of original features. Each tree $t_i, 1 \leq i \leq n$ represents a transformation of the ith feature. The fitness function is the accuracy of the kNN classifier in the transformed data.

In [34], the wrapped classification algorithm is a generalized linear machine (GLIM), whose accuracy is used as the fitness function. Each individual is a tree that uses arithmetic operations to encode a combination of the original features. In this approach, the original features are not used by the classifier; the evolved feature is the only one taken into account in classification. In [35], the same authors extend their previous work as follows. Three types of classifiers are used: GLIM, kNN, and the maximum likelihood classifier. Individuals use the same codification as in the previous reference, but in addition, each individual is associated with one type of classifier. This way, the GP does not only choose the best constructed feature, but also chooses the most suitable type of classifier. Another improvement to the previous approach is the inclusion of a size penalty in the fitness function, so that fitness is based on accuracy, but with a preference for smaller individuals.

In [32], GP is wrapped around C4.5, and the accuracy of the decision tree induction algorithm is used as the fitness measure as usual. In this approach, each individual is a forest containing a certain number (chosen by the user) of trees, each one encoding a new constructed feature. A portion of these trees is used as an elitist repository, where the best individuals are kept. Individuals to be preserved in the elitist subforest are chosen on the basis of a utility measure given by the number of times the constructed feature is used by the learning algorithm (C4.5). In [25] and [26], a similar approach is proposed, in which each individual is a forest composed of a number of trees chosen by the user, and the fitness measure is given by the accuracy of the classifier. Here, however, elitism is not used, and the base classifier is a simple linear perceptron or a centroid-based classifier.

In [27], GP is wrapped around a Bayesian classifier, and the fitness function is given again by the accuracy of the classifier. Each individual is a tree representing a constructed feature.

A hybrid GP/GA system is presented in [33], where GP is used for feature construction and a GA performs a selection from the constructed features. In this paper, several base classifiers have been wrapped by the GP feature constructor (in different runs), namely C4.5, kNN, and a Bayesian classifier. Each individual is a forest containing n trees, where n is the number of attributes in the original dataset, and each tree is able to represent a constructed feature. Classification accuracy is used as the fitness measure. Once the GP feature construction is done, a GA is applied, also as a wrapper, to make a selection from the constructed features.

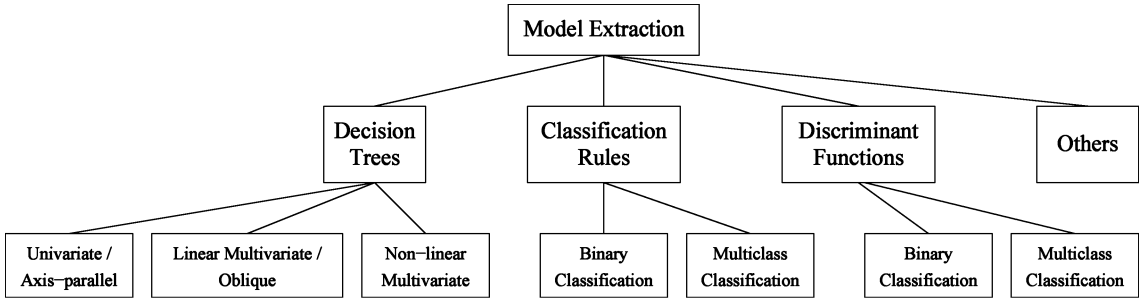

Fig. 2. Model extraction with GP.

The system proposed in [29] employs a coevolutive GP (CGP) algorithm as a wrapper over a Bayesian base classifier, using classification accuracy as the fitness function. CGP is an extension of GP in which several populations are maintained and employed to evolve solutions cooperatively. A population maintained by CGP is called a subpopulation and it is responsible for evolving a part of a solution. A complete solution is obtained by combining the partial solutions from all the subpopulations. In this paper, each individual in each subpopulation is a tree encoding a constructed feature. When the evolutionary process ends, the best individual from each subpopulation is selected, and these best individuals are joined in a new feature vector.

B. Feature Selection

In [22], GP is employed to achieve a selection of features. Each individual is a tree encoding a classifier represented as a discriminant function (see Section V-C), and classification accuracy is used as the fitness function. The authors capitalize on the implicit feature selection ability of GP classifiers (see Section IV) by applying GP for classification in a two-stage scheme. First, a certain number of GP runs are carried out, each one resulting in a best-of-run classifier. Then, in the second stage, the GP is run again using only the features most frequently used in the best classifiers obtained in the previous stage. However, the goal of the first stage is just to discover the most interesting features, and the classifiers obtained are discarded. The final classifier is induced in the second stage using the features selected in the first stage.

V. GP for Model Extraction in Classification

The basic idea that lies behind the application of GP for inducing classifiers consists in making each individual represent a classifier or a part of a classifier and defining a fitness function to measure its quality, so that the evolutionary process leads to a high quality final classifier.

Most of the papers published related to GP and classification focus on the application of GP to model extraction, that is, the induction of classifiers. For this reason, a great number of references will be reviewed in this section, which is further divided, first according to the way in which classifiers are represented, and second according to some other criteria when the number of references is considerable. Fig. 2 shows the structure of the points addressed in this section.

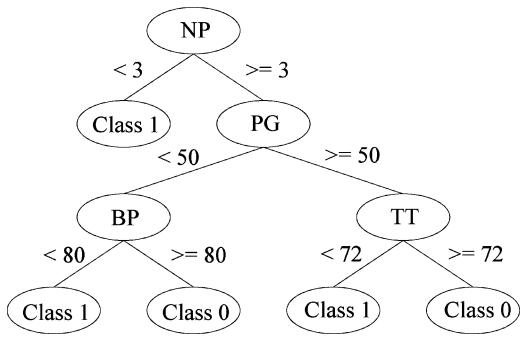

Fig. 3. Example of a decision tree.

A. GP for Extracting Decision Trees

Decision trees [15] are one of the most frequently used representations for classifiers. A vast amount of literature has been devoted to this form of classification. However, we must keep in mind that tree structures are also the preferred encoding scheme for GP individuals. Thus, the application of GP to evolve decision trees seems to be an obvious approach. Within such a framework, it is customary to make each individual of the GP population encode a decision tree, and that is how most of the works included in this section proceed.

A decision tree contains zero or more internal nodes and one or more leaf nodes. All internal nodes have two or more child nodes. All internal nodes contain splits, which test the value of an expression of the attributes. Arcs from an internal node t to its children are labeled with distinct outcomes of the test at t. Each leaf node has a class label associated with it. Fig. 3 represents an example decision tree for the Pima dataset from the UCI repository (see [36]), a binary classification problem.[1] Fig. 4 shows a GP individual corresponding to this decision tree. There are three main types of decision trees based on how the feature space is partitioned.

1) Univariate or axis-parallel decision trees. This type of decision tree carries out tests on a single variable at each nonleaf node. Their mode of splitting the data is equivalent to using axis-parallel hyperplanes in the feature space.
2) Linear multivariate or oblique decision trees. A linear combination of features can be tested at internal nodes. The tests are geometrically equivalent to hyperplanes at an oblique orientation to the axis of the feature space.

[1] NP, PG, BP, and TT are attribute names in this dataset.

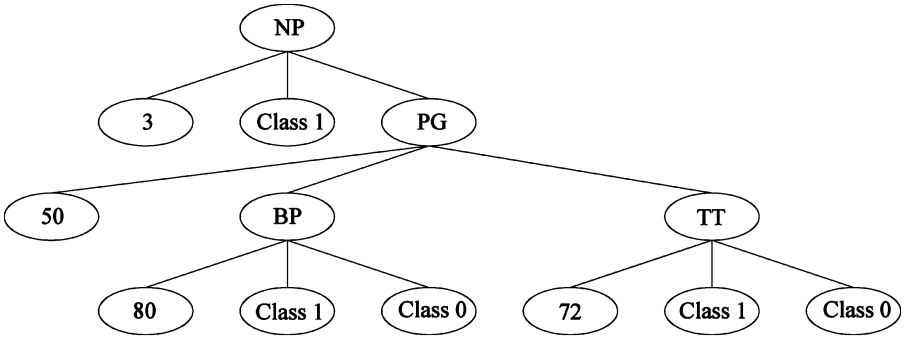

Fig. 4. GP individual representing a decision tree.

TABLE III
APPLICATIONS OF GP-INDUCED DECISION TREES

Application area	References
Medicine	[37]–[39]
Finance	[40]
Software engineering	[41]
Character recognition	[42]–[45]
Benchmark	[44], [46]–[54]
Synthetic	[55], [56]

3) Nonlinear multivariate decision trees. Nonlinear combinations of features can appear inside internal nodes. They perform a nonlinear partitioning of the feature space.

When GP is applied, it is natural to use individuals encoding some kind of structure including arithmetic operations, and both linear and nonlinear combinations of features can be readily represented. As a result, oblique and nonlinear tests can be evolved as easily as univariate tests. The relation of this point with feature construction should be evident; because of the nature of GP, when it is applied in order to induce classifiers, an implicit feature construction is usually done.[2] As we will see in the following sections, these considerations are relevant not just to decision trees, but also to other representation formalisms. Table III lists the applications of the systems here reviewed.

1) Axis-Parallel Decision Trees: In [56], a basic and straightforward system is described, in which axis-parallel decision trees are evolved using classification accuracy as the fitness function. Axis-parallel decision trees are also evolved in [46], using a fitness function combining accuracy and tree size. The previous proposal is extended in [47] by means of hybridizing GP with simulated annealing. This technique is used to avoid premature convergence of the GP algorithm. A new evolved individual is kept, if it has better fitness than others in the population, otherwise it is accepted with a probability proportional to a temperature parameter, which decreases as evolution proceeds.

Some works focus on obtaining compact (comprehensible) axis-parallel decision trees. In [42], tree size is not included in the fitness function which is given by classification accuracy. Instead, tree complexity is considered in the selection of individuals. During evolution, individuals with higher fitness are selected, but when several individuals have equal fitness, the smallest ones are preferred. Another approach to obtain small accurate decision trees with a simple accuracy-based fitness is proposed in [43]. The key fact behind this approach lies in the relation observed between the sizes of the training set and the induced classifiers: usually, the larger the training set, the larger the decision tree obtained. Therefore, the divide and conquer technique is applied by splitting the training set into several subsets and then evolving different subtrees, which will be later combined to make up the final decision tree. In [45], the same idea of using a subset of the training data in order to obtain smaller trees is used, but in a different way, employing a C4.5/GP hybrid system. The fitness function is computed again as classification accuracy. C4.5 is used to initialize the population. A certain number of training examples is randomly selected and used to induce a (small) decision tree with C4.5. This process is repeated for each of the individuals in the initial population, and then the GP algorithm performs the evolutionary process using the complete training set.

Another work involving the induction of small axis-parallel decision trees is presented in [37]. The fitness function considers both classification accuracy and tree size. A similar approach is followed in [40], where a fitness function combining accuracy and size is employed. But in this work smaller trees are enforced not only by the fitness function, but also by the use of two operators specifically designed to simplify trees: elimination and merge. Another similar proposal is described in [38], but instead of using specific genetic operators, small trees are enforced by means of a tree pruning method.

In [55], we find an analytical study on the influence of the fitness function and mutation operator on the quality of the decision trees induced, taking into account both classification accuracy and tree size. One of the main points of interest when constructing decision trees with GP is how to obtain accurate and parsimonious classifiers, since simple classifiers are more comprehensible to humans. This concern is addressed by many researchers, not only when evolving decision trees, but also when other representation formalisms are employed.

Fuzzy (axis-parallel) decision trees are evolved in [49]. Continuous attributes are fuzzified for convenience. Three fitness measures are employed in this paper: standard fitness is the usual classification accuracy fitness measure; precision fitness is computed from the membership values of training data; size fitness, given by the number of nodes in the tree. However, only

[2]When the function set includes arithmetic operators.

precision fitness is employed during the evolutionary process. To select the best individual in an evolutionary run, standard fitness is considered first, and in the case of a tie, the smaller individual is chosen.

In [44], MO is applied in order to optimize two goals: classification accuracy and tree size. Four different MO methods are proposed by the authors. Smaller trees are further enforced by the application of an editing (pruning) method aimed at removing redundant nodes, which can be combined with any of the MO methods.

In [48], an MOGP system designed to optimize two goals is described, but these goals are not fixed. The authors make a general proposal, allowing for any pair of goals to be considered. In this paper, a cost-sensitive point of view is adopted [57], and related pairs of goals are suggested, such as false negative rate/false positive rate, sensitivity/specificity, or recall/precision, but any other pair of goals could be employed.

In [41], MOGP is applied once again. The system proposed here has been specifically designed to tackle a particular classification problem: software quality classification. The purpose of the system is to discriminate between fault-prone and non fault-prone software modules. Classification performance is evaluated by a modified expected cost of misclassification (MECM) measure, but the software quality assurance team has limited resources to inspect suspicious software modules. MO is applied to accommodate three goals: to minimize the MECM, to obtain a number of modules predicted as fault-prone, that is, equal to the number of modules that can be inspected with the available software quality improvement resources, and to minimize the size of the decision tree model.

2) Oblique Decision Trees: The potential of GP to include any kind of operation into individuals is used in [50] in order to evolve oblique decision trees. A simple accuracy-based fitness function is employed. Oblique decision trees are also evolved in [51]. The fitness function takes into account classification accuracy and applies two size penalty factors, one for the number of nodes and the other for the depth of the tree.

Oblique decision trees are constructed in [52] with MO considering two goals: to maximize classification accuracy and to minimize the number of nodes. Furthermore, some other techniques are employed to improve performance, such as limited error fitness, employed to reduce running time, and fitness sharing, used to promote diversity in the population.

3) Non-Linear Decision Trees: Quite a different approach is presented in [53]. Each individual is a tree representing a node, instead of a decision tree. Each node can encode arbitrarily complex conditions, including linear and nonlinear combinations of attributes. The decision tree is constructed in a top–down way, starting by the root and descending through successive levels while new nodes keep improving the overall fitness or until a maximum number of levels is reached. The fitness function combines classification accuracy and generalization ability.

An MOGP system is also employed in [39] in order to evolve fuzzy nonlinear decision trees. Continuous attributes are fuzzified, and MO is applied to optimize two goals: accuracy, measured as the area under the receiver operating characteristic (ROC) curve [58], and complexity, measured as the size of the tree.

A population of decision trees is evolved in [54], allowing trees that are as general as needed to be constructed, so that even nonlinear multivariate trees could be evolved. Fitness is based on classification accuracy, with a penalty for size, but in an unusual way, since smaller individuals are penalized in this proposal.

B. GP for Learning Rule-Based Systems

Rules are a simple and easily interpretable way to represent knowledge [1], [59]. A rule has two parts, the antecedent and the consequent. The rule antecedent contains a combination of conditions for the predicting attributes. Typically, conditions form a conjunction by means of the AND logical operators, but in general any logical operator can be used to connect elemental conditions, also known as clauses. The rule consequent contains the value predicted for the class. This way, a rule assigns a data instance to the class pointed out by the consequent if the values of the predicting attributes satisfy the conditions expressed in the antecedent; hence a classifier is represented as a rule set.

Usually, classification rules employ only simple conditions, where an attribute is compared with a value from the corresponding domain by means of some relational operator, resulting in univariate linear classifiers. In some occasions, conditions comparing the value of one attribute to the value of another attribute are also allowed, leading to linear multivariate classifiers. More complex conditions are rarely employed.

Two main approaches exist for individual encoding when evolving rule-based classifiers.

1) In the *individual=rule* approach, each individual encodes a rule, and hence, some method must be used to construct the classifier from the rules evolved, once the evolutionary process finishes.
2) In the *individual=rule set* approach, each individual represents a complete classifier, that is, a rule set, and so the classifier will be the best of the individuals evolved.

Another consideration is often taken into account when constructing rule-based classifiers. Usually, different approaches are employed depending on the number of classes to be distinguished.

1) In binary classification, there are just two classes to be distinguished. This usually leads to simpler algorithms.
2) Multiclass algorithms are able to distinguish any number of classes, and are hence, more general, but can also be more complex.

Binary and multiclass problems are sometimes handled in different ways. Some methods are restricted to binary problems, while other algorithms can cope in a seamless way with any number of classes. Since any n-class problem can be reduced to $n-1$ binary classification problems, methods limited to binary problems can still be applied to multiclass problems. However, methods able to directly handle any number of classes are generally more suitable, because they can be applied more easily and readily. The following review of works is categorized based

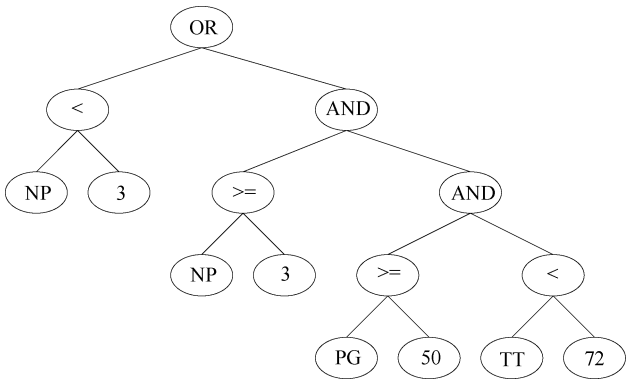

Fig. 5. GP individual representing a classification rule.

TABLE IV
APPLICATIONS OF GP-INDUCED CLASSIFICATION RULES

Application area	References
Biology	[60]–[62]
Medicine	[63]–[70]
Finance	[71]–[74]
Object recognition	[75]
Character recognition	[76]
Benchmark	[54], [77]–[86]

the type of classification task (binary/multiclass). Fig. 5 represents a GP individual corresponding to the antecedent of the rule shown as follows for the Pima dataset

$$\begin{aligned} \text{IF} \quad & ((\text{NP} < 3) \\ \text{OR} \quad & ((\text{NP} \geq 3) \text{ AND } (\text{PG} \geq 50) \text{ AND } (\text{TT} < 72))) \\ \text{THEN} \quad & \text{Class } 1. \end{aligned}$$

Table IV lists the applications of the systems here reviewed. We can note the substantial number of references dealing with medical issues. The reason is that in Medicine, it is especially important to understand the rationale behind the classification that the system outputs. Since rules give an easily interpretable representation, they are often preferred in medical appli cations.

1) Binary Classification: Binary classification is carried out in the proposal described in [82] and [83], where each individual in the population encodes a rule, and a single rule is obtained as the final result of the evolutionary process. Simple univariate conditions are employed. Since just two classes must be distinguished, the instances not covered by that rule are assigned to the other (default) class. This paper employs a self-adapting fitness function. Fitness is measured as classification accuracy, but in a weighted way. Each instance from the training set has a weight associated, used to weigh the partial fitness of each instance. Weights are also evolved, in such a way that instances which are harder to classify have their weights increased.

In [75], like in the work described in [82] and [83], a single rule for one of the classes is evolved, considering the remaining class as the default class. This system allows the use of conditions comparing the value of two attributes, in addition to the usual simple univariate conditions which compare the value of an attribute with a constant. This way, linear multivariate classifiers can be obtained. The fitness function is based on classification accuracy. A similar system is proposed in [72], but only simple univariate conditions are employed in the antecedents of the rules.

Fixed-length individuals are evolved in [61] with the purpose of obtaining simpler and more comprehensible binary classifiers. Each individual encodes a rule. An accuracy-based fitness function is employed.

In [60], a single rule is evolved in each running, and the system is run twice, one time for each of the classes. The fitness function is based on the rms error of prediction, and includes a penalty factor for size. Arithmetic operations and the comparison of attributes and their combinations are allowed to appear in the conditions, leading to nonlinear multivariate classifiers.

An *individual=rule* approach is also employed in [63], where both univariate and linear multivariate conditions are allowed. When the evolution ends, the final classifier is formed by the best n rules, where n is a value chosen by the user, so that classifiers with several rules per class can be obtained. The fitness function is based on ROC/AUC measures.

Fuzzy rules for binary classification are evolved in [73]. A classifier that can contain several rules is constructed, but all the rules predict the same class; the other one is the default class. Each individual encodes a rule, employing univariate conditions. A modal evolutionary process is followed, in which each run results in a rule. All items correctly classified by this rule are removed from the training set and the system is restarted. This process continues until every instance of the objective class is described by a rule. MO is used in order to simultaneously optimize four fitness functions, three of them related to the accuracy of the rule and the remaining one measuring its size. Several parallel runnings of the systems are performed, each one with different values for the parameters controlling the evolutionary process. Each running produces a complete classifier, and a committee-based ensemble method is used to obtain a final classifier from these rule sets.

A comparison of several variant GP-evolved classifiers for binary classification following the *individual=rule set* approach can be found in [70]. Both crisp and fuzzy rules combining univariate as well as multivariate conditions are evolved.

A two-stage evolutionary approach is proposed in [65]. In the first stage, a hybrid GP/GA method is employed to evolve a good pool of rules. Each individual represents a rule, or to be more precise, a part of a rule, since GP is employed to evolve the part of the antecedent involving categorical attributes and a GA is used to evolve the conditions concerning numerical attributes. A run is performed for each of the classes considered, but a set of rules (for the same class) is selected in each running. Token competition is used in order to obtain a proper covering of all the data instances, and the fitness function measures accuracy. The second phase of the system uses the pool of rules produced

by the first phase as its basis. A simple evolutionary algorithm is employed; each individual represents a rule set, and several subpopulations are formed. If the number of rules in the candidate pool is n, then there will be n subpopulations and the ith subpopulation will be evolved to optimize the rule set containing i rules. At the end of the evolution, each subpopulation outputs its best candidate rule set, which will compete (based on classification accuracy) with the best rule sets generated by other subpopulations to obtain the final optimal rule set.

The system described in [74] is aimed at the problem of classification with imbalanced data. Data is said to be imbalanced when some classes heavily outnumber others. The problem with imbalanced data arises because learning algorithms tend to overlook less frequent classes, paying attention just to the most frequent ones and ignoring the minority classes. Usually, the minority class is the positive class, that is, the class that the end-user is interested in, but this is the class most difficult to characterize, due to its limited number of instances. The approach followed in this paper consists in generating rules only for the positive class. Each individual in the population can represent a set of rules. Since only conjunctive antecedents are allowed in the rules, when a logical OR is found in a tree, it separates one antecedent from another; this way, rules are extracted from all the individuals in the population. A repository is kept in order to construct the final classifier. Each extracted rule is added to the repository if its precision is greater than a given threshold and if it is different than rules already in the repository. Conditions comparing the value of two attributes are allowed, so that linear multivariate classifiers can be obtained.

2) Multiclass Classification: Multiclass classification is addressed in [67]. A single rule is evolved in each run of the system, and n runs are performed for a n-class classification problem. This way, the final classifier has a single rule for each class. The fitness function combines three terms, two of them (specificity and selectivity) are used to gauge the predictive ability of the rule, and the third one measures its complexity. Simple univariate conditions are employed. A similar approach is followed in [77].

In the same vein as the previous references, the system presented in [78] tackles multiclass classification performing a different running for each of the classes to be distinguished. However, several differences can be noted with respect to the systems described above. Each individual represents a rule, but a niching mechanism (namely token competition) and elitism are employed in such a way that a set of rules, all of them predicting the same class, are obtained at the end of the evolutionary process, so that the final classifier can have several rules for each of the classes. Simple univariate conditions are also used in this paper. An accuracy-based fitness function is employed.

Each individual represents a rule in the system described in [71]. It employs a fitness function based on sensitivity and specificity and simple univariate conditions in order to construct classifiers with several rules per class. A similar system aimed at evolving fuzzy rules is proposed in [64].

In the proposal described in [79], based on the *individual=rule* approach, a separate run is performed for each of the classes to be discriminated. When an evolutionary run finishes, the individuals with the highest fitness values found in any generation are saved. Finally, a selection strategy builds the final classifier from these sets. It begins by forming a set choosing the best rule in each set corresponding to a particular class. Then, the remaining rules are progressively evaluated in order to determine whether they will be included in the set, checking if the predictive quality of the rule set has improved. The fitness function is a combination of precision and support. The search is biased toward comprehensible solutions in the selection process. Each generation, individuals are ranked probabilistically according to one of three possible criteria: support, precision or comprehensibility. A probability is assigned to each criterion. These probabilities self-adapt as the evolutionary process runs.

In [87], a hybrid GA/GP system in which each individual represents a rule is described. Classifiers containing several rules per class can be constructed. The fitness function takes three terms into consideration: support, confidence, and rule size. This system allows the use of conditions comparing the value of two attributes; in this way, linear multivariate classifiers can be obtained.

Linear multivariate classifiers containing several rules per class, evolved employing an *individual=rule* approach with the aid of the token competition niching method, are used in [80]. A fitness function based on the concepts of support and confidence is employed. This approach is also employed in other studies which extend the previous proposal. In [68], a two-stage process is followed. In the first phase, evolutionary programming (EP) is used to evolve a Bayesian network, which represents an overall structure of the relationships among the attributes. The Bayesian network provides knowledge in itself, but it is also used as a basis to define the grammar to be used by the GP algorithm in the second stage. This system is further extended in [69], where a GA is added to the EP and the GP. The GA is used to discretize the continuous attributes. Similar proposals can be found in [76] and [81].

The system described in [62] employs a fitness function based on classification accuracy with a penalty for size. Each individual encodes a rule set. Arithmetic operations and the comparison of attributes and combinations of attributes are allowed to appear in the conditions, leading to nonlinear multivariate classifiers.

The hybrid GA-P system [88] is applied for evolving fuzzy rules in [84]. A coevolutionary approach is employed, where the GP algorithm evolves a population of rule sets while the GA evolves the parameters defining the membership functions of the attributes used in the antecedents of the rules. Each individual encodes a complete classifier, in which only univariate conditions are allowed. A classifier can contain several rules per class.

The system proposed in [85] evolves fuzzy classification rules for multiclass problems. In this paper, the numeric attributes are fuzzified, and a coevolutive approach is followed, in which a GP algorithm and an evolution strategy (ES, another kind of evolutionary algorithm [89]) are run simultaneously in a cooperative way. Each individual of the GP encodes a rule set and each individual of the ES represents the membership functions corresponding to the fuzzified continuous attributes. In continuation, we focus on the GP algorithm. Each individual in the

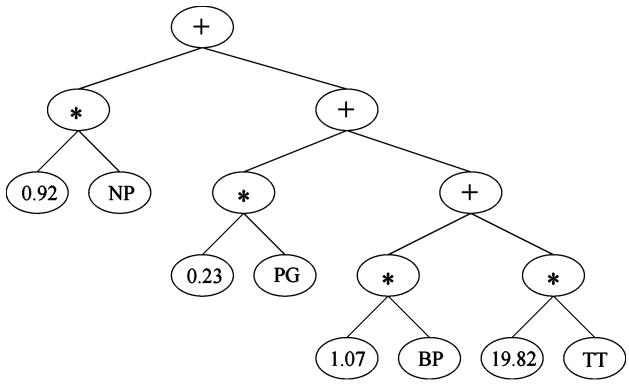

Fig. 6. GP individual representing a discriminant function.

population represents a rule set, but all the rules of each of the individuals predict the same class, and hence, the system must be run once for each of the classes to be distinguished. Simple univariate conditions are used in the antecedents of the rules. The fitness function measures accuracy based on sensitivity and specificity. Size is not included in the fitness function, but it is considered in the selection of individuals. When individuals are selected to breed the next generation, individuals with higher fitness are selected, but when several individuals have equal fitness, the smallest ones are preferred.

The system proposed in [86] evolves classifiers in which several rules per class can exist applying an *individual=rule set* approach. Simple univariate conditions are used in the antecedents of the rules. The fitness function is based on classification accuracy. A similar approach for the evolution of fuzzy rules is employed in [54].

A hybrid *individual=rule/individual=rule set* approach for multiclass classification is followed in [66]. An individual can contain multiple classification rules, subject to the restriction that all its rules have the same consequent. The population consists of a set of individuals of this type. When the evolutionary process finishes the best individual for each class is used to form the final classifier, thus, several rules can exist for each class. Simple univariate conditions are allowed. The fitness function combines three terms, sensitivity and specificity are used to gauge the accuracy of the rule, and a term based on size measures its complexity.

C. GP for Learning Discriminant Functions

Discriminant functions are another formalism for representing classifiers. A function is a mathematical expression in which different kinds of operators are applied to the attributes of a data instance that must be classified. This way, a single output value is computed from the operations performed on the values of the attributes. The value computed by the function indicates the class predicted. Usually, this is accomplished by means of a threshold or set of thresholds. For binary classification problems, a single function is enough; if the output value is greater than a given threshold, the example is assigned to a certain class, otherwise it is assigned to the other one. Usually, the threshold is zero, so that a positive output indicates a particular class, while a nonpositive value corresponds to the other class. Fig. 6 represents a GP individual corresponding to the following discriminant function for class 1 of the Pima dataset

$$0.92^{*}\mathrm{NP} + 0.23^{*}\mathrm{PG} + 1.07^{*}\mathrm{BP} + 19.82^{*}\mathrm{TT}$$

TABLE V
APPLICATIONS OF GP-INDUCED DISCRIMINANT FUNCTIONS

Application area	References
Biology	[90]
Medicine	[91]–[99]
Oceanography	[100]
Finance	[101], [102]
Engineering	[28], [103]–[106]
Communications	[107]–[109]
Software engineering	[110]
Object recognition	[111]
Character recognition	[112]
Generic image classification	[113]–[122]
Signal classification	[123]
Benchmark	[4], [124]–[137]
Synthetic	[124]

For multiclass problems, two basic approaches can be followed. As we have seen in Section V-B, one possibility is to consider a n-class classification problem as $n-1$ binary problems, so that $n-1$ or n binary threshold functions, like the ones described above, would be used in order to discriminate the n classes. The other option is to use just a single function to distinguish all the classes. Now $n-1$ threshold values are needed. These thresholds will determine n intervals, and each interval will be assigned to a particular class; this way, the predicted class will depend on the interval which the output value belongs to. Though these methods are pervasively followed in the literature, some other approaches have been devised, as we will see as we review the published works.

The obvious approach for evolving this kind of classifier with an evolutionary algorithm like GP is to have a population in which each individual encodes a discriminant function. The function set used by the GP algorithm will determine the kind of operations that the function can perform on the data. In the following review, works are categorized based on the number of classes to be distinguished (binary/multiclass).

Table V lists the applications of the systems reviewed in this section. In this case, we can find a significant number of references related to image classification and other related pattern recognition issues. The reason is that the representation given by discriminant functions is very akin to the kind of mathematical operations and transformations usually applied to signal processing.

1) Binary Classification: The most obvious and simple approach to tackle binary classification problems is to evolve a population of threshold functions, using classification accuracy as the fitness measure, and to choose the most fit of the evolved individuals as the final classifier. This basic approach can be found in [91], [92], [95], [103], [114], and [115].

One of the first works on the application of GP for evolving classification functions is [111], where binary classification is

addressed by means of the evolution of a single zero-threshold discriminant function. Two different ways of measuring fitness are proposed. The first one consists in the application of an MO approach in order to simultaneously optimize two goals: classification accuracy (correct classification rate) and the a *posteriori* entropy of class distributions. The second fitness function is the false alarm or FP rate. Another work involving MO is [107], where three objectives are considered: tree complexity, misclassification cost, and an approximation to the Bayes error in the 1-D projected space.

The goal of Cavaretta and Chellapilla [127] is to find out if the principle of Occam's razor holds for classifiers. A single-threshold function is evolved. Two fitness functions are considered; the first one is computed as the correct classification rate and the second one is a distance-based fitness, in which the difference between the real and the predicted output is computed for each instance. Since the goal is to test the suitability of Occam's razor, both fitness measures can be penalized with a parsimony factor, in such a way that fitness decreases with the number of nodes of the tree which encodes an individual. Similar works in which binary classification is addressed by evolving a single-threshold function, with a fitness function based on accuracy including a size penalty, can be found in [93], [105], [116], [117], and [126].

Other approaches to binary classification have been proposed, for example, in [106]. A single-threshold function is evolved employing a fitness function composed of two terms, the first one is classification accuracy, and the second one is a measure of certainty, which refers to the probability of misclassification in future examples, different from the ones in the training set; that is, certainty is considered in this paper as a measure of the generalization ability of the classifier.

The system described in [94] is run several times. In each running, one best-of-run discriminant function is obtained. A pool of functions is constructed after several rounds. After training, data instances are classified by a majority voting on the outputs of the individual classifiers.

A system in which several validation sets are used to assess the quality of the induced classifier is described in [110]. The system works by following three stages: training, multiple dataset validation and voting. In the training phase, 40 runs are performed, obtaining 40 models, from which the six best models are kept. In the validation phase, each of the six selected models is applied to five different datasets to validate their generalization abilities. The datasets are different, but all of them are similar and related to a same domain. After all the validations end, the model with the best validation performance is selected. The above process is repeated six times for a chosen test dataset. Each time one of the remaining six datasets is selected as a fit dataset, while the other five datasets are used in the validation phase. After six repetitions, six best models, each generated in one repetition, can vote the classification for the given test dataset.

Two fitness functions specially tailored for coping with class imbalance are presented in [124]. Each individual in the population is a single-threshold discriminant function.

In [90], each individual is a discriminant function encoded as a program in a specific programming language that will be run on a certain virtual machine. The fitness function is based on the distance between the real and predicted output, but some penalties are considered. There is a penalty for size, and some other penalties can be applied when the program performs certain illegal operations. Machine code individuals are evolved to carry out binary classification in [101], where a fitness function combining the mean square error and the correct classification rate is used, and in [102], where the fitness function is based on the distance between the real and the predicted class.

Three active learning methods are compared in [125]. The goal of active learning is to perform dynamic selection of examples while the evolutionary process is running, in order to alleviate the computational overhead of GP. The basic idea is to select the instances that are more difficult to classify for the individuals that are being evolved. The fitness function employed in this proposal is based on the distance between the real and the predicted class.

MO and coevolution are applied in [130] in order to evolve a set of classifiers (a Pareto front) while achieving an efficient fitness evaluation. Two populations are coevolved, one population in which each individual represents a threshold discriminant function and another population in which each individual encodes a subset of the training set. When the evolution of the function population ends, a set of nondominated classifiers is obtained, rather than a single classifier. Each classifier gives a classification output for a given example, and a voting system is applied to obtain the final classification. The purpose of the population of training subsets is to obtain a training subset smaller than the original training set, since the smaller the training set, the shorter the time needed to compute fitness. Since the training subset coevolves with the classifiers, the fitness of a training subset depends on the performance of the classifiers with that data. Classification accuracy is used as the fitness measure for the discriminant functions.

In [113], each individual is a single-threshold discriminant function. The main contribution of this paper is the inclusion of restricted loops in the function set. These loops include parameters that are also evolved by the system.

2) Multiclass Classification: One of the first works involving multiclass classification with GP-evolved discriminant functions is [131], where an n-class problem is converted into n binary problems. The system is run once for each of the classes to be distinguished. In each running, a single-threshold discriminant function is evolved for a particular class. Classification accuracy is used as the fitness measure. A similar approach is described in [100] with a fitness function measuring the overlapping between the class outputs given by the classifier.

Multiclass classification is carried out in [118] by means of a set of single-threshold discriminant functions and a simple voting scheme. A set of functions is evolved, which can result in several functions for a given class. Each of these functions discriminates one class from the rest. When an instance is to be classified, each of the functions gives its output and the final prediction is obtained through voting.

The system described in [128] addresses multiclass classification by constructing n binary classifiers, so that it has to be run once for each of the classes. This system employs a

layered multipopulation GP framework. A layer is composed of a number of populations. The result of evolving each population is a discriminant function. These functions transform the training set to construct a new training set, performing a feature generation process, so that new constructed features are added to the original ones when passing to the next layer. The last layer has a single population, so that a single classifier is finally constructed from the best partial solutions given by the previous layers. To put it another way, each layer evolves a part of the final discriminant function, which is progressively constructed layer-by-layer. The fitness function is based on classification accuracy. A similar method is presented in [96], but here the fitness function is based on sensitivity and specificity, and two feature selection methods are added to the system.

A GP-based approach to multiclass classification in which an integrated view of all classes is taken is proposed in [129]. This system is able to construct a complete classifier in a single run. Each individual is a multitree structure made up by n trees, n being the number of classes. Each of these n trees ($T_i, 1 \leq i \leq n$) encodes a threshold function for a particular class. The system considers that a data instance x belonging to class i is correctly classified, if $T_i(x) \geq 0$ and $T_j(x) < 0$, for all $j \neq i$. The fitness function is computed as the classification accuracy. A similar system evolving a multiple-threshold discriminant function is described in [97], where a fitness function based on the sum of squared errors is employed.

In [98] and [122] multiclass classification is addressed by evolving a multiple threshold discriminant function. This function distinguishes n classes by means of $n-1$ threshold values. These thresholds determine n intervals, and each interval is assigned to a particular class; this way, the class predicted will depend on the interval that the output value belongs to. Different approaches can be devised from this basic idea. The simplest approach, static class boundary determination, consists in fixing the boundary values at manually chosen points. In [119], two methods designed to dynamically determine thresholds during the evolutionary process are presented. These approaches are further improved in [120].

In the proposal presented in [121], each individual represents a multiple-threshold discriminant function, but instead of using the single best evolved individual in the population, this approach uses multiple evolved programs to perform classification; the best evolved individuals are selected to form the classifier. It is assumed that the behavior of a program classifier is modeled using multiple Gaussian distributions each of which corresponding to a particular class. The distribution of a class is determined by evaluating the program according to the examples of that class in the training set. If the curves representing these distributions are well separated, this means that the classes are being correctly distinguished, and hence, the degree of overlap is used as the fitness measure. The classifier is formed by several discriminant functions. The probability that an instance belongs to a certain class is computed based on the values of the probability density function for a given class according to the different discriminant functions, and the instance is assigned to the class with the highest probability.

The two approaches for multiclass classification, constructing a single classification function or n binary classifiers are compared in [112]. This paper addresses the problem of hand-written digit recognition. When a single function able to discriminate all the classes is evolved, this function directly outputs the numeric value of the predicted class, since each class is an integer digit. The other option consists in running the system as many times as there are classes to be distinguished, each run evolving a single-threshold discriminant function for a particular class. In both cases, the fitness function is based on classification accuracy.

Two methods for multiclass classification are proposed in [108]. In the first method, a single function is evolved. When the evolutionary process ends, the best evolved individual is chosen as the classifier, and it is applied to each of the instances in the training set, so that a set of values is obtained for each class. When a new instance has to be classified, the classification function is applied to it and it is assigned to the class that has the output obtained in its value set. If the output obtained does not appear in any of the value sets, the instance is assigned to the class which has the value that is nearest to the output obtained. The second method consists in choosing the p best evolved individuals (p can be different from the number of classes). Each function will output a corresponding class with a confidence factor for each sample. The outputs obtained are combined to compute the membership factor for each class, and the instance is assigned to the class with the highest membership factor. Both methods employ a distance-based fitness function designed to measure interclass discrimination and separation. A similar system is described in [123]. Other studies comparing different approaches to multiclass classification with discriminant functions can be found in [104] and [132].

A single best-of-run individual is evolved as the final classifier in [4] and [133]. Each class has an associated numerical value. A data sample is classified as belonging to the class that is nearest to the output value computed by the discriminant function evolved. Fitness is computed as the classification accuracy.

In [134], attributes are fuzzified, and multiclass classification is achieved by evolving a single-threshold discriminant function for each class. The fitness function is based on the distance between the real and predicted output. A similar approach is proposed in [135], but using rough sets instead of fuzzy sets. Similar proposals can be found in [136] and [137].

A system for the evolution of discriminant functions is described in [99], where each individual in the population is a string representing a C program, which maintains an output array having a position for each of the classes to be predicted. The program computes n output values, one for each of the classes; the predicted class is the one with the highest output value. The fitness function combines the mean square error and the correct classification rate. A similar system is described in [109], but here the n-class problem is treated as n binary problems, running the system n times.

D. GP With Other Representations

In the previous sections, we have reviewed the studies that employ GP to evolve classifiers in their more common forms

TABLE VI
APPLICATIONS OF OTHER KINDS OF GP-EVOLVED CLASSIFIERS

Application area	References
Biology	[138], [139]
Finance	[140]–[142]
Character recognition	[143]
Generic image classification	[144]
Benchmark	[54], [142], [145]–[154]
Synthetic	[154]

(decision trees, classification rules, and discriminant functions). Though less common, there are many other formalisms that can be used to represent classifiers. In this section, we focus on publications dealing with GP-evolved classifiers where other representations are employed. Table VI lists the applications of the systems reviewed in this section.

A hybrid system in which the classifier is represented using both classification rules and discriminant functions is described in [140]. GP is used in a two-stage evolutionary system for binary classification. In the first stage, rules are evolved individually. An individual represents a rule, and each rule corresponds to a particular class. One or several rules can be evolved for each class. The training set is reduced as rule generation proceeds, so that the instances covered by existing rules are eliminated from the training set. Only simple univariate conditions are allowed to appear in the rules. In the second stage, a single-threshold discriminant function is evolved from the reduced training set (the data which do not satisfy any rule or satisfy more than one rule). The fitness function used in both stages is based on the distance between the predicted and the real class. Once the hybrid classifier has been constructed, the procedure followed to classify a new instance consists in first applying the rule set. If the instance is covered by some rule, it is classified in the class predicted by that rule, otherwise, it is classified by the discriminant function.

ANNs have been extensively used for classification and other purposes [155]. An ANN consists in a group of processing units or neurons highly interconnected in a layered system. GP is employed in [138] to evolve neural networks. Each individual in the population is a tree representing the architecture (structure and parameters) of the network. The fitness function is the classification accuracy of the network throughout the training data. In a similar way, GP has been applied in [54] and [141] for the evolution of neural networks, where again each individual in the population represents a network architecture.

SVMs have emerged in recent years as a popular approach to classification [156], [157]. SVMs transform the data into a new space where the data is linearly separable, and then build a hyperplane which divides examples so that examples of one class are all on one side of the hyperplane while examples of the other class are all on the other side. A kernel function is used to facilitate the construction of such a hyperplane. GP has been applied to the evolution of kernel functions for SVMs. The basic idea consists in evolving a population in which individuals are kernel functions, whose fitness is computed by constructing a SVM from it and measuring its quality. This setting is followed in [146] and [147], where GP is used to evolve kernels. In [145], each individual represents a multiple or hybrid kernel (a combination of kernels given by a mathematical expression). Multiple kernels are similarly evolved in [148], but here a generalization measure called bound of generalization error is used as fitness, and a hybrid GP/evolution strategy (ES) method is employed. The GP part is used to optimize the structure of the multiple kernel, while the ES optimizes the parameters of the subkernels. Another work dealing with the evolution of multiple kernels by means of GP can be found in [149]. Here, a boosting technique is applied to concentrate evolution on the most difficult objects to classify. The hybrid kernels obtained at each boosting round participate in the training of SVMs, which are combined into a final classifier, applying a weighted voting approach.

In a similar way to a rule set, a classifier can also be represented as a set of queries. Queries are usually expressed in a database management system query language like SQL, but other options can be taken into consideration. This approach is followed in [150], where GP is employed to evolve a population where each individual is a SQL query. The conditions that are allowed to appear in queries are comparisons of an attribute with a constant, but also comparisons of one attribute with another one, and conditions using SQL specific operators, like EXITS or ALL. Fitness is evaluated with a function which includes a term for accuracy, another term for complexity and another one for the shape of the tree, that can be used for encouraging certain kinds of queries. A population of SQL queries is evolved using GP in [158]. A set of $n-1$ queries is used to distinguish n classes. All the rules are obtained in a single run using elitism and some niching method. Simple univariate conditions are employed in this proposal. The framework proposed in the previous work is directly applied in [142] to binary classification. Another SQL-based system for multiclass classification employing GP is described in [159]. Here, each individual is a complete classifier, in which several rules per class are allowed. Fitness is computed as the accuracy rate. A special kind of classification, text classification, is addressed in [143]. Binary classification is carried out in this system, where each individual in the population is a query expressed in Lucene, a text search language. Specific Lucene operators are used to build the queries, and fitness is based on precision and recall.

The kNN algorithm is an instance-based learning method in which a dataset is used as a reference to classify new instances, with the help of a suitable distance measure. In order to classify a new data instance, its kNNs are found, the number of instances in each class is counted for that subset of k, and the example to be classified is assigned to the class with a higher count. These class counts can be weighted with the purpose of trying to avoid the adverse effect caused by outliers. GP is used in [144] to evolve such kind of weighted counts. Each individual in the population is a tree encoding a mathematical expression which provides a weighted count of the kNNs. The system proposed is aimed at binary classification. The fitness function is based on the ROC curve.

Several proposals for the application of EAs to optimize the performance of the kNN method are presented in [154]. First a GA is used to select the training set. MO is used in order to

simultaneously minimize the size of the training set and maximize classification accuracy. Another proposal consists in using GP to evolve the distance measure used by the kNN algorithm. MO is applied again in order to maximize the classification rate while minimizing the complexity of the distance measure. The next idea consists in combining the previous ones in a cooperative coevolutive setting: fitness evaluation for the training set selection species is done by computing the accuracy rate for each individual using the best distance measure of the previous generation; similarly, the fitness for the proximity measure species is computed from the accuracy rate for each individual using the best training set of the previous generation. Finally, a third species that will compete with the two previous ones is added. A separate fitness evaluation dataset is used to evaluate the fitness of individuals. The third species is employed to select the instances in this fitness evaluation set, and the goal of its fitness function is to minimize the accuracy rate. The effect achieved with this third species is to enhance performance, since fewer instances are used to compute fitness, and to attain a better generalization capability.

Kernels can be applied not only to SVMs, but also to any learning method involving a distance measure. The kernel nearest neighbor method (kernel-NN) [160] is a variation of the kNN algorithm in which a kernel is applied to the distance function. GP is employed in [151] in order to evolve kernels for kernel-NN. The fitness function measures the margin for a correct classification that the prototype set (the reference data instances used for classification) provides. The training data is usually used both as the prototype set and as the dataset to compute the fitness of an individual. However, when GP is wrapped around a kernel-NN classifier to search for the most suitable kernel for a particular dataset, this setting would lead to prohibitive running times. The method proposed applies coevolution to tackle this problem by reducing the number of prototypes used for classification and reducing the fitness case subset considered during each generation. Three species are employed: the first species includes the GP kernels; the second species includes the prototype subset subject to a cooperative coevolution with the GP kernels; the third species includes the fitness case subset subject to a competitive coevolution with the GP kernels. The prototype species is evolved to find good prototypes that maximize the fitness of the GP kernels. The fitness case species is evolved to find hard and challenging examples that minimize kernel fitness. Both prototype and fitness cases species are evolved using an evolution strategy [89].

A novel approach to multiclass classification with GP is described in [139]. GP is employed to evolve what is known as variable predictive models (VPMs). A VPM is a mathematical model that expresses the value of a variable as a function of one or a set of the other variables. In this work, each possible pair of variables $\{X_i, X_j\}$ is considered, where X_i is the output (predicted) variable and X_j is the input (predicting) variable. This way, a GP-evolved VPM is constructed for each X_i as a function of each of the other variables X_j, performing a separate GP run for each VPM. The best VPM_i from all the VPM_{ij} is kept for each variable X_i. The root mean squared error is used as the fitness measure. This set of computations is performed separately with the data instances of each class, so that, in the end, the best VPMs are kept for each variable and each class. When a new data instance has to be classified, each feature is predicted with the corresponding VPM for each of the classes, and the instance at issue is assigned to the class whose VPMs give a prediction of the values of the features that is closer to the real value.

Multiclass classification is addressed in a rather unusual way in [152]. The approach proposed is motivated by the market-based Hayek model [161], [162], an artificial market model. The model, based on auctions, returns a set of individuals that decomposes the problem by way of a bidding process. Each individual in the population has a certain wealth and an associated action (a class assignment, in classification problems), and can bid in an auction for certain data instances. The individual making the highest bid wins the instance and has to pay for it, but will be rewarded if the action is suited to that instance (if the predicted and the real classes match). The amount bid is determined by a mathematical expression computed according to the values of the attributes, evolved by the GP algorithm. The classifier is a set of individuals that can bid for instances in a particular class, and fitness is measured on the basis of the wealth of individuals. A similar system is described in [153], but here coevolution is applied in order to evolve a training subset along with bidding classifiers, and the concept of wealth is not employed.

TABLE VII
APPLICATIONS OF GP-BASED ENSEMBLES

Application area	References
Biology	[166]–[168]
Medicine	[169]–[171]
Pharmacology	[172]
Communications	[173]
Benchmark	[174], [175]
Synthetic	[171], [176]

VI. GP FOR LEARNING ENSEMBLE CLASSIFIERS

In this section, we focus on a particular kind of postprocessing technique aimed at obtaining the maximum from the classification models produced by learning algorithms: ensemble or team methods. The basic idea behind this approach is to improve the quality of the prediction by means of using not a single classifier, but a group of them, each one providing a possibly different output. Ideally, different base classifiers in an ensemble capture different patterns or aspects of a pattern embedded in the whole range of data, and then through ensembling, these different patterns or aspects are incorporated into a final prediction. For this reason, diversity is very important for ensembles. Bagging and boosting are two well-known ensemble methods [163]–[165].

Two main issues have to be addressed when an ensemble approach is employed, which are how to generate diverse base classifiers and how to combine base classifiers, and GP has been applied to both issues, as we will see shortly. Table VII lists the applications of the systems reviewed here.

The application of GP in order to combine the predictions of several classifiers is addressed in [172]. First, several partitions

of the original training data are formed and the base classifiers are obtained from these datasets by means of different learning algorithms, e.g., linear classifiers, naïve Bayes, C4.5, and ANNs. Each of the classifiers obtained is translated into a mathematical expression, that can be used as building blocks for GP. Each GP individual is a tree in which base classifiers are combined by means of mathematical and logical operators. Several classifiers of the same kind or classifiers of different types can be combined to obtain an ensemble. The fitness function used in these works is based on the ROC curve.

A parallel implementation of a system for evolving ensembles is described in [171]. First, the population is subdivided into demes (subpopulations assigned to different nodes) which, in turn, are subdivided into teams of individual programs. The fitness function combines the mean square error and the correct classification rate. A simpler sequential system is described in [166], in which an accuracy-based fitness function is used. GP is employed to evolve diverse base classifiers.

The goal of the system proposed in [168] is to enhance accuracy and reduce performance fluctuation of classifiers produced by GP. A pool of classifiers is independently evolved in parallel. Each base classifier is a linear genome machine. An ensemble is randomly formed by selecting individuals from the pool. The ensemble is accepted if the individuals' faults occur independently of each other. Otherwise the process is repeated until an acceptable ensemble is obtained. The behavioral diversity of the classifier is measured by the expected error rate of combined individuals.

GP is used in [173] to evolve a set of diverse base classifiers. Here, the focus is on the efficient construction of classifiers from huge volumes of data. When the training data is massive, the learning process can be too lengthy. An obvious approach in such a situation is to sample the training data and use just a subset of it to construct the classifier, but important data can be missed this way, resulting in poor classification performance. This drawback can be alleviated using an ensemble. Small subsets from a large dataset are used to build a number of base classifiers, and then these base classifiers are combined. Binary classification is addressed in this paper. Each base classifier is a single-threshold discriminant function, evolved in a GP run using classification accuracy as the fitness function. The combining mechanism used is simple majority voting, that is, predictions of the majority of base classifiers is set as the ensemble prediction.

A system for binary classification is described in [169]. Bagging is used in order to obtain different training sets. A single-threshold discriminant function is evolved from each of the training sets, using classification accuracy as the fitness measure. A subset of diverse classifiers is selected from this pool to make the final classifier. Diversity is measured by comparing the structure of the classification rules instead of output-based diversity estimating. Nine different fusion methods for combining the results from the base classifiers are compared.

Binary classification is also addressed in [170]. Each of the base classifiers is a single-threshold discriminant function, evolved with a fitness function combining accuracy and size. Diverse base classifiers are obtained here by first applying the k-means clustering algorithm [177] to the training data, followed by a feature selection algorithm applied to obtain the most relevant features from each cluster. This way, each of the base classifiers is evolved throughout the training data but with different feature subsets.

A boosting-based distributed ensemble system for streaming data is presented in [176]. There is streaming data when new data continuously flows into a dataset at high speed; in these circumstances, a tremendous and rapidly increasing amount of data has to be processed. An island distributed model is employed. Each node receives its own flow of data, and evolves one of the base classifiers, represented as decision trees, using classification accuracy as the fitness function. Once the ensemble classifier has been built using a subset of the streaming data, the ensemble is used to classify the new incoming data, and there is no need to retrain unless a concept drift is detected (a change in the nature of the data). Concept drift is detected using a fractal-based self-similarity measure. When the performance of the ensemble falls behind a certain threshold, retraining is performed on the next available chunk of streaming data. A similar boosted distributed system is also employed in [174], where the k-means clustering algorithm [177] is employed in order to choose the most diverse and fittest base classifiers. After a round of generations, the fittest individual from each of the clusters formed is selected from each of the subpopulations. In addition, a diversity-based pruning mechanism is applied to reduce the number of base classifiers in the ensemble.

A comparison of different methods for evolving ensembles of GP classifiers is carried out in [175]. There are two common methods to evolve teams of genetic programs, island and team approaches. Island approaches produce teams of strong individuals that cooperate poorly, while team approaches produce teams of weak individuals that cooperate strongly. A new approach known as orthogonal evolution of teams (OET) is proposed in this paper. This method overcomes the weaknesses of island and team approaches by applying evolutionary pressure at both team and individual levels during selection and replacement. The base classifiers are multiple-threshold discriminant functions, suitable for multiclass classification. The final classification is obtained by a weighted voting. Different fitness measures are used for base classifiers and for teams of classifiers, both based on classification accuracy.

Multiclass classification is tackled in [167] by dividing an n class problem into n binary problems, and performing n runs, one for each of the classes. Here, an individual encodes a small-scale ensemble system containing a set of trees known as a subensemble (SE). Each of the trees in a SE represents a single-threshold discriminant function. The output of each SE is computed by means of a weighted voting scheme. The best SE for a given class is obtained after each running, together with a weight, which will be employed to obtain the final classification by combining the output of all the best SEs.

VII. Pitfalls, Benefits, and Guidelines

The application of GP to classification tasks has its advantages and drawbacks. In this section, we analyze the problems

associated with the use of GP for classification, but we will also see that these problems can be circumvented. Besides, we pinpoint the specific advantages of applying GP for classification purposes. Finally, we give some guidelines on how to employ GP for classification.

A. Pitfalls on the Use of GP-Based Classifiers

The main problem of GP is its computational cost, primarily due to the computation of the fitness of individuals, which has to be repeatedly evaluated through the evolutionary process. Fitness evaluation is particularly costly when evolving classifiers, since it implies the application of the classifier encoded by each individual in the population to all the data instances in the training set. Training times on the order of seconds [79], [121], [122], [130], [134], minutes [51], [78], [80], [125], [137], hours [68], [85], [153], or even days [63], [123] have been reported in the literature reviewed in this paper.

Another problem of GP is that caused by introns or junk code in individuals. Introns are pieces of code in an individual produced by the evolutionary process that does not affect the fitness of the individual, so that they can be viewed as junk or useless code. Introns are problematic because there is usually an exponential growth of junk code in individuals as the evolutionary process goes on, causing the situation known as bloating, which means that GP individuals grow uncontrollably until they reach their maximum size, filled with useless code. This situation usually leads to run stagnation and poor results, and bigger individuals imply even higher training times, worsening computational cost problems of GP. These are problems caused by introns in GP in general, but another adverse effect must be pinpointed from the specific point of view of classification with GP. Comprehensibility is one of the factors which determine the quality of the induced models, and it can decrease drastically when the classifier is swamped with introns. But despite the obvious problems caused by introns, there is some evidence that junk code can also have beneficial effects, protecting good building blocks in the individuals from the destructive effects of crossover. Different mechanisms have been proposed to palliate the adverse effects of introns, such as using nondestructive crossover operators, applying a parsimony pressure to individuals by including a size penalty in the fitness function or employing fitness functions that change throughout the evolutionary process.

Finally, another difficulty of GP arises from the great number of parameters that must be defined in order to have a working system. Some of these parameters are population size, number of generations, probabilities of application of different genetic operators or maximum size of individuals, to name a few. Usually, a series of preliminary runs are required in order to tune all these parameters. Nonetheless, this drawback of GP can be considered only as a secondary one, in comparison with computational cost and bloating.

While these problems, mainly computational cost and introns, are of great importance in every application of GP, the papers reviewed here generally do not address these issues; only a few cases afford them marginal attention. This situation is understandable because the authors focus on their main goal: to find better ways of applying GP in order to obtain better classifiers. Besides, many solutions have been proposed in GP specific literature to cope with the well-known problems of this technique,[3] and all of these techniques can be applied to any GP-based system. Therefore, researchers concerned with GP classification concentrate on obtaining better classifiers, and consider that the existing solutions for GP specific problems can be added to the classification system later on.

Although most of the works reviewed here pay no attention to specific GP drawbacks, there are some exceptions that we would like to enumerate here. Parallel and distributed implementations are probably the most obvious approach to decrease training time, and are applied in [63], [73], [90], [109], [158], [168], [171], [174], and [176]. The use of a parallel demetic framework has been found to improve running time even when implemented on a single processor machine, as explained in [99]. Incremental learning is applied in [134] and [137] in order to decrease training time. Incremental learning consists of beginning the evolutionary process using just a small subset of the training set, and adding training instances as the evolutionary process goes on, until completing the whole training data. MO and coevolution are applied in [130] to evolve a set of good classifiers while achieving efficient fitness evaluation. Two populations coevolve, one population in which each individual represents a discriminant function and another population in which each individual encodes a subset of the training set. The purpose of the population of training subsets is to obtain a training subset smaller than the original training set, since the smaller the training set, the shorter the time needed to compute fitness. A reduction in training time is reported in [128] as a consequence of the use of the multi-population GP framework proposed. Three different active learning methods are proposed in [125] in order to improve training time. The basic idea behind active learning consists in using the fitness of the models induced through the evolutionary process to guide the selection of samples used for training. Although the high computational overhead of GP makes scalability more difficult, scalable GP-based classification systems can be constructed, and some papers are specifically devoted to the classification of large-scale datasets with GP [125], [173]. Scalability is sought in [151] by coevolving the prototype subset and the fitness case subset together with the kernel for a kernel-NN classifier. In fact, GP has been applied even to the classification of streaming data (data continuously flowing into a dataset at a high speed) [176].

With regard to introns, there are a considerable number of papers that address bloating as a side effect. As we have seen, comprehensibility is one of the desired features of classifiers, and one of the factors related to comprehensibility is the size of the model; the bigger the classifier, the harder to interpret for humans. One of the most usual ways of promoting smaller models is the inclusion of a size penalty in the fitness function, but this parsimony pressure also prevents the presence of introns. In this way, the efforts aimed at enhancing comprehensibility

[3]Proposals for speeding up the evolutionary process and protecting it from introns can be found in [3].

are also beneficial with respect to bloating. This kind of size penalty can be found in many of the papers reviewed [37], [40], [62], [90], [102], [104], [105], [116], [117], [126], [127], [139], [143]. On some occasions, size penalties are proposed with the specific purpose of fighting bloat [51], [60], [107]. A selection mechanism specially designed to tackle bloat, known as lexicographic tournament selection is employed in [147]. A pruning operator is employed in [85] to diminish bloating effects. The system proposed in [171] removes introns when computing the fitness of individuals. The same intron removal algorithm is applied in [61] with a twofold purpose: to remove introns and obtain simpler and more comprehensible classifiers. Nondestructive genetic operators are employed in [104] to avoid bloating. The main issue addressed by [98] is the proposal of a new online simplification method to remove redundant code, based on algebraic equivalences and hashing techniques.

B. *Benefits of the Use of GP-Evolved Classifiers*

At this point, we have seen that GP has some drawbacks that can, however, be circumvented in different ways. But, why should we use GP to solve a classification problem? Classification is one of the most researched problems in computer science and dozens of methods for classifier induction exist. This leads us to question how well GP compares with other classification techniques and what the reasons could be for choosing GP.

Regarding classification accuracy, many of the papers that we review include performance comparisons of classifiers obtained by means of GP and some other techniques like decision tree algorithms, neural networks, SVMs, instance-based learning, or others. A total of 124 papers have been reviewed in Section V. There are 66 out of these 124 papers in which GP is compared with some other techniques in terms of accuracy; usually, comparisons are carried out over several different datasets. In these comparisons, there are 174 cases in which GP is the algorithm obtaining the best accuracy, while in 144 cases, GP is beaten by some other technique. This means that in a 54.72% of the comparisons, GP is the best performing method. Summarizing, it can be said that GP-based classifiers generally compare quite well with the ones induced by other algorithms. Usually, we can find that GP-based classifiers reach the highest accuracy for some datasets, equal the best competitor for some others and in some cases are inferior to some other ones. This should not be a surprise if we take into consideration the no free lunch theorem [178], [179]. But even when GP-based classifiers are beaten by some other algorithm, most of the times the differences are small, giving a performance near to the best one.

At this point, we have seen that GP can be used to construct classifiers with competitive accuracy, but its main drawback is still its computational cost. Although some other methods like neural networks can have comparable training times, other techniques like C4.5 will probably always be much faster. So, it is time to look at the specific advantages of using GP for classification.

We think that the key feature of GP that makes it so interesting is its flexibility. This flexibility yields different advantages. As we have seen, GP individuals can use any representation formalism: decision trees, classification rules, discriminant functions, etc. But GP can also evolve the setting for a neural network, an SVM, a kNN classifier or any other conceivable classification mechanism. It is because of its flexibility that the same tool (GP) can be applied for different classification tasks, at a preprocessing stage, for classifier induction and also for postprocessing purposes, therefore, the researcher or practitioner does not need to be familiar with many different techniques. Flexibility also means that a GP-based system can be tailored, adapted and tuned to every possible need; for example, any fitness function, designed according to the requirements of a given problem, can be plugged into the system. In a similar way, a great variety of different genetic operators, evolutionary frameworks, diversity promotion techniques, and many other evolutionary mechanisms are at our disposal, therefore, we can choose the ones that best fit our needs.

Another advantage of GP lies in its ability to perform explicit and automatic feature selection and extraction. This avoids the need to perform explicit preprocessing and gives additional benefits, as we will see in short order.

Although GP is comparable in accuracy to any other method, accuracy is not the only factor that determines the quality of a classifier. There are some other desirable properties for a classifier, and GP offers important advantages here. Interpretability is one of the factors that affect classification quality. Some classification techniques are considered as black box methods because they focus on accuracy but provide classifiers that are difficult for people to understand. Neural networks, statistical methods, SVMs and kNN are some techniques that have been traditionally considered as black box methods. White box methods comprise algorithms that produce interpretable and comprehensible classifiers, like decision trees and classification rules. As we see, one of the main factors determining interpretability is the representation formalism. Most classification algorithms are limited to one specific type of representation, like decision tree algorithms, rule induction algorithms, neural network algorithms, etc. However, GP individuals can be adapted to use any of these representations. This way, a more interpretable formalism, like rules, can be chosen when necessary. Usually, interpretability receives less attention than accuracy, and the number of papers concerned with this issue is reduced. Only 13 out of the 124 papers reviewed in Section V report interpretability comparisons between GP and other techniques, with interpretability measured in terms of classificator size. In 60 cases, GP provides the more interpretable classifiers, while in 30 occasions GP is beaten by some other method. This yields a 66.67% of cases in which GP is the best performing method.

But GP has some other advantages regarding comprehensibility. One of the factors which determines comprehensibility is complexity, usually measured in terms of size. GP can be configured to promote the evolution of simple (more comprehensible) classifiers, for example by setting a maximum tree depth or including a size penalty in the fitness function. Another factor related to classification quality is interestingness. Although this factor is very difficult to measure quantitatively, GP can clearly contribute to it. The analysis of the classifiers evolved can provide interesting insights which help researchers

better understand the problem at hand. The automatic feature selection performed by GP indicates which attributes are more relevant in a particular domain, and the features constructed point out interesting relationships between variables. Some of the papers reviewed stress this kind of added value produced by GP as a side effect [23], [63], [90], [93], [99], [102], [106].

Finally, we would like to highlight the potential of one of the GP variants: grammar-based GP (GBGP) [80], [180]. GBGP was devised as a solution to circumvent the closure problem. It is based on the utilization of grammars to specify a language, which individuals must adhere to. GBGP makes use of genetic operators that take the grammar into account, in such a way that every individual generated is always guaranteed to be legal with respect to the grammar. The use of a grammar allows us to declaratively specify many features of the classifiers to be evolved. This way, we can control different factors related to comprehensibility, for example specifying the kind of conditions that can appear in rule antecedents, or how these conditions can be connected (in disjunctive normal form, for example). But with a grammar, we can also set up an *individual=rule* or *individual=rule set* representation, and thus, many facets of evolvable classifiers. We feel that the possibility of using grammars is an advantage of GP because it enhances GPs flexibility and expressive power.

C. Guidelines for GP-Based Classification

GP is a very flexible and powerful technique that can be employed in different ways to carry out classification tasks. As we have seen, different GP variants and representation formalisms can be applied. In this section, we give some advice aimed at guiding researchers on the best options to use depending on the characteristics of the classification problem at issue. However, we have to point out that these hints are general and empirically derived rules of thumb; these guidelines must not be considered rigidly or dogmatically.

With regard to preprocessing, GP is better suited for feature extraction than feature selection, since the capability of GP to incorporate into individuals mathematical operations that are applied to data variables fits very naturally with the task of feature extraction. However, we have seen that an implicit feature selection process is performed when GP is applied, and feature extraction is readily carried out with GP in a straightforward and natural way, there is no need to incorporate a separate specific feature construction algorithm. This way, when feature selection and/or extraction is necessary for the classification problem at hand, GP could be very well suited.

One of the most important decisions to be taken when considering the application of GP to classification is the representation formalism to be employed. On some occasions, interpretability is the main concern, and an easily interpretable classifier can be preferred even at the expense of a slightly lower accuracy. This is usually the case in Medicine, where an explanation for the reasons behind a particular diagnosis is required. In these circumstances an interpretable representation must be employed. Rules are usually considered as the most interpretable representation. GP-evolved classification rules have been extensively applied to medical problems, as we have seen in Table IV (see Section V-B).

Independently of the representation formalism employed, another way to foster interpretable classifiers consists in including a penalty factor in the fitness function. Smaller models are easier to understand.

On other occasions, accuracy is the main goal and interpretability is sacrificed. For example, in financial domains, a slight increase in accuracy can imply a higher income percentage. Discriminant functions can be a more suitable representation in these circumstances. Discriminant functions are often employed also in applications related to image classification and other related pattern recognition issues (see Table V in Section V-C). The reason is that the representation given by discriminant functions is very akin to the kind of mathematical operations ad transformations usually applied to signal processing. However, we must take into consideration that, because of the nature of discriminant functions, this representation is well suited to numerical data, but not to categorical data.

Turning our attention to efficiency issues, an ensemble approach can be useful when learning from large datasets, since each base classifier can be induced from different disjoint datasubsets. A parallel implementation of GP can also obviously be beneficial when dealing with large datasets.

To sum up, why to use GP for evolving a classifier? We cannot give a precise and absolute answer to this question. The interested researcher should take into consideration the different drawbacks and advantages of GP, together with the guidelines that we have compiled, and ponder whether the GP technique could be suitable or not for the problem at hand. When the researcher feels that some of the advantages of GP can provide a valuable benefit or fits naturally to the particular needs and characteristics of the problem at issue, then GP should probably be given a try.

VIII. Concluding Remarks

This paper presents a survey of GP for classification. We begin by providing a brief analysis of the cardinal points in the two areas concerned: GP and classification. This provides us with the background context needed to understand the works reviewed, and serves as a guideline to categorize and sort relevant literature.

A considerable number of papers have been published on the application of GP for classification. Most of these papers focus on the core step of classifier induction, a task that can be accomplished by evolution using GP. But the great flexibility of GP allows it to be applied not only to the construction of classifiers; some preprocessing and postprocessing tasks aimed at enhancing the quality of classification have also been addressed employing GP.

We would like to end up by pointing out some of the research lines that could merit further attention in the future.

One of the main drawbacks of GP is its high training time, which worsens when compounded with the need for dealing with the huge amounts of data often found in classification problems. It is necessary to delve into the possibilities available to make

GP training as efficient as possible, like parallel and distributed GP, for example.

In classification, the quality of induced models is determined by several features. Literature has been basically focused on accuracy, while interpretability has attracted only marginal attention. GP can be of great value at this point. Grammars, for instance, could be readily employed to shape interpretable classifiers.

In relation with the previous point, a way of taking into consideration several factors affecting classification quality is the application of multiobjective optimization. A deeper research on the opportunities of applying MO techniques to classification with GP can be beneficial.

The combination of different techniques allows us to make the most of several methods, leveraging on their strengths and avoiding their drawbacks. The flexibility of GP makes it possible to combine it with very diverse methods. But the combination of GP with some other technique is not the only option; GP can be employed as a mechanism to combine different algorithms. A structure indicating how different base classifiers are combined can be optimized by evolution with GP.

Although classification is one of the most researched tasks in computer science, some new issues still arise, like the problem of learning from imbalanced data [181], [182]. Data is said to be imbalanced when some classes differ significantly from others with respect to the number of instances available. The problem appears because learning algorithms tend to overlook less frequent classes, leading to poor classification rates in the less frequent classes. It is necessary to explore the opportunities that GP can offer to deal with imbalanced data.

Classification is a basic task which can serve many different purposes, like credit scoring, bankruptcy prediction, medical diagnosis, and so many more. Many of these applications have been extensively addressed in the existing literature, but new areas of application are arising, like streaming data or web mining, and the application of GP can be of great value in these new contexts.

REFERENCES

[1] J. Han and M. Kamber, *Data Mining—Concepts and Technique* (The Morgan Kaufmann Series in Data Management Systems), 2nd ed. San Mateo, CA: Morgan Kaufmann, 2006.

[2] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Reading, MA: Addison-Wesley, 2005.

[3] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, *Genetic Programming—An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. San Mateo, CA/Heidelberg, Germany: Morgan Kaufmann/dpunkt.verlag, 1998.

[4] M. Oltean and L. Diosan, "An autonomous GP-based system for regression and classification problems," *Appl. Soft Comput.*, vol. 9, no. 1, pp. 49–60, Jan. 2009.

[5] J. C. Bezdek, S. Boggavarapu, L. O. Hall, and A. Bensaid, "Genetic algorithm guided clustering," in *Proc. 1st IEEE Conf. Evol. Comput.*, Orlando, FL, Jun.1994, pp. 34–39.

[6] L. Jie, G. Xinbo, and J. Li-cheng, "A GA-based clustering algorithm for large data sets with mixed numeric and categorical values," in *Proc. 5th Int. Conf. Comput. Intell. Multimedia Appl.*, Xi'an, China: IEEE, Sep. 2003, pp. 102–107.

[7] I. D. Falco, E. Tarantino, A. D. Cioppa, and F. Fontanella, "An innovative approach to genetic programming-based clustering," in *Proc. 9th Online World Conf. Soft Comput. Ind. Appl.* (Advances in Soft Computing Series, 34)., Berlin, Germany: Springer-Verlag, Sep./Oct. 2004, pp. 55–64.

[8] Y. Liu, T. Özyer, R. Alhajj, and K. Barker, "Cluster validity analysis of alternative results from multi-objective optimization," in *Proc. 5th SIAM Int. Conf. Data Mining*, Newport Beach, CA, 2005, pp. 496–500.

[9] R. Alhajj and M. Kaya, "Multi-objective genetic algorithms based automated clustering for fuzzy association rules mining," *J. Intell. Inf. Syst.*, vol. 31, no. 3, pp. 243–264, Dec. 2008.

[10] M. Lyman and G. Lewandowski, "Genetic programming for association rules on card sorting data," in *Proc. Genet. Evol. Comput. Conf.*, Washington, DC: ACM, Jun. 2005, pp. 1551–1552.

[11] W. M. Spears, K. A. De Jong, T. Bäck, D. B. Fogel, and H. de Garis, "An overview of evolutionary computation," in *Proc. Eur. Conf. Mach. Learning* (Lecture Notes in Computer Science, 667). Berlin, Germany: Springer-Verlag, Apr. 1993, pp. 442–459.

[12] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. of Michigan Press, 1975.

[13] C. M. Bishop, *Pattern Recognition and Machine Learnin* (Information Science and Statistics Series). Berlin, Germany: Springer-Verlag, 2006.

[14] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "The KDD process for extracting useful knowledge from volumes of data," *Commun. ACM*, vol. 39, no. 11, pp. 27–34, Nov. 1996.

[15] S. K. Murthy, "Automatic construction of decision trees from data: A multi-disciplinary survey," *Data Mining Knowl. Discov.*, vol. 2, no. 4, pp. 345–389, 1998.

[16] G. L. Pappa, A. A. Freitas, and C. A. A. Kaestner, "Attribute selection with a multi-objective genetic algorithm," in *Proc. 16th Brazilian Symp. Artif. Intell. - Adv. Artif. Intell.* (Lecture Notes in Computer Science Series, 2507). Porto de Galinhas/Recife, Brazil: Springer-Verlag, Nov. 2002, pp. 280–290.

[17] J. R. Cano, F. Herrera, and M. Lozano, "Evolutionary stratified training set selection for extracting classification rules with trade off precision-interpretability," *Data Knowl. Eng.*, vol. 60, no. 1, pp. 90–108, Jan. 2007.

[18] R. C. Holte, "Very simple classification rules perform well on most commonly used datasets," *Mach. Learning*, vol. 11, pp. 63–91, 1993.

[19] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York: Wiley, Jun. 2001.

[20] C. A. C. Coello, G. B. Lamont, and D. A. V. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problem* (Genetic and Evolutionary Computation Series), 2nd ed. Berlin, Germany: Springer-Verlag, 2007.

[21] H. Liu and H. Motoda, *Feature Extraction, Construction and Selection: A Data Mining Perspectiv* (The Springer International Series in Engineering and Computer Science Series, 453).. Berlin, Germany: Springer-Verlag, 1998.

[22] R. A. Davis, A. J. Charlton, S. Oehlschlager, and J. C. Wilson, "Novel feature selection method for genetic programming using metabolomic ^{1}H NMR data," *Chemometrics Intell. Lab. Syst.*, vol. 81, no. 1, pp. 50–59, Mar. 2006.

[23] M. L. Raymer, W. F. Punch, E. D. Goodman, and L. A. Kuhn, "Genetic programming for improved data mining—Application to the biochemistry of protein interactions," in *Proc. 1st Annu. Conf. Genetic Program. 1996*, Cambridge, MA: MIT Press, Jul., pp. 375–380.

[24] H. Guo and A. K. Nandi, "Breast cancer diagnosis using genetic programming generated feature," *Pattern Recog.*, vol. 39, no. 5, pp. 980–987, May 2006.

[25] C. Estébanez, J. M. Valls, R. Aler, and I. M. Galván, "A first attempt at constructing genetic programming expressions for EEG classification," in *Proc. 15th Int. Conf. Artif. Neural Netw.: Biol. Inspirations* (Lecture Notes in Computer Science Series, 3696). Warsaw, Poland: Springer-Verlag, Sep. 2005, pp. 665–670.

[26] C. Estébanez, J. M. Valls, and R. Aler, "GPPE: A method to generate ad-hoc feature extractors for prediction in financial domains," *Appl. Intell.*, vol. 29, no. 3, pp. 174–185, Dec. 2008.

[27] X. Tan, B. Bhanu, and Y. Lin, "Fingerprint classification based on learned features," *IEEE Trans. Syst., Man, Cybern. C*, vol. 35, no. 3, pp. 287–300, Aug. 2005.

[28] H. Guo, L. B. Jack, and A. K. Nandi, "Feature generation using genetic programming with application to fault classification," *IEEE Trans. Syst., Man, Cybern. B*, vol. 35, no. 1, pp. 89–99, Feb. 2005.

[29] Y. Lin and B. Bhanu, "Evolutionary feature synthesis for object recognition," *IEEE Trans. Syst., Man, Cybern. C*, vol. 35, no. 2, pp. 156–171, May 2005.

[30] M. Muharram and G. D. Smith, "Evolutionary constructive induction," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 11, pp. 1518–1528, Nov. 2005.

[31] K. Neshatian and M. Zhang, "Genetic programming and class-wise orthogonal transformation for dimension reduction in classification problems," in *Proc. 11th Eur. Conf. Genet. Program.* (Lecture Notes in Computer Science Series, 4971), Naples, Italy: Springer-Verlag, Mar. 2008, pp. 242–253.

[32] K. Krawiec, "Genetic programming-based construction of features for machine learning and knowledge discovery tasks," *Genet. Program. Evol. Mach.*, vol. 3, no. 4, pp. 329–343, Dec. 2002.

[33] M. G. Smith and L. Bull, "Genetic programming with a genetic algorithm for feature construction and selection," *Genet. Program. Evol. Mach.*, vol. 6, no. 3, pp. 265–281, Sep. 2005.

[34] J. Sherrah, R. E. Bogner, and A. Bouzerdoum, "Automatic selection of features for classification using genetic programming," in *Proc. Australian New Zealand Conf. Intell. Inf. Syst.*, Piscataway, NJ: IEEE, Nov.1996, pp. 284–287.

[35] J. R. Sherrah, R. E. Bogner, and A. Bouzerdoum, "The evolutionary pre-processor: Automatic feature extraction for supervised classification using genetic programming," in *Proc. 2nd Annu. Conf. Genet. Program.*, San Mateo, CA: Morgan Kaufmann, Jul.1997, pp. 304–312.

[36] P. Murphy and D. Aha. (1994), "UCI repository of machine learning databases," Dept. Inf. Comput. Sci., Univ. California, Irvine, CA [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[37] J. K. Estrada Gil, J. C. Fernández-López, E. Hernández-Lemus, I. Silva-Zolezzi, A. Hidalgo-Miranda, G. Jiménez-Sánchez, and E. E. Vallejo-Clemente, "GPDTI: A genetic programming decision tree induction method to find epistatic effects in common complex diseases," in *Proc. 15th Int. Conf. Intell. Syst. Molecular Biol., 6th Eur. Conf. Comput. Biol. (Suppl. Bioinf.)*, Vienna, Austria, Jul. 2007, pp. 167–174.

[38] C.-S. Kuo, T.-P. Hong, and C.-L. Chen, "An improved knowledge-acquisition strategy based on genetic programming," *Cybern. Syst.*, vol. 39, no. 7, pp. 672–685, Oct. 2008.

[39] E. M. Mugambi, A. Hunter, G. Oatley, and R. L. Kennedy, "Polynomial-fuzzy decision tree structures for classifying medical data," *Knowl. Based Syst.*, vol. 17, no. 2–4, pp. 81–87, May 2004.

[40] C.-S. Kuo, T.-P. Hong, and C.-L. Chen, "Applying genetic programming technique in classification trees," *Soft Comput.*, vol. 11, no. 12, pp. 1165–1172, 2007.

[41] T. M. Khoshgoftaar and Y. Liu, "A multi-objective software quality classification model using genetic programming," *IEEE Trans. Rel.*, vol. 56, no. 2, pp. 237–245, Jun. 2007.

[42] M. Shirasaka, Q. Zhao, O. Hammani, K. Kuroda, and K. Saito, "Automatic design of binary decision trees based on genetic programming," presented at the 2nd Asia-Pacific Conf. Simul. Evol. Learning, Canberra, Australia, Nov. 1998.

[43] T. Tanigawa and Q. Zhao, "A study on efficient generation of decision trees using genetic programming," in *Proc. Genet. Evol. Comput. Conf.*, San Mateo, CA: Morgan Kaufmann, Jul. 2000, pp. 1047–1052.

[44] S. Haruyama and Q. Zhao, "Designing smaller decision trees using multiple objective optimization based GPs," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, vol. 6, Piscataway, NJ: IEEE, Oct. 2002, p. 5.

[45] S. Oka and Q. Zhao, "Design of decision trees through integration of C4.5 and GP," in *Proc. 4th Jpn.-Australia Joint Workshop Intell. Evol. Syst.*, 2000, pp. 128–135.

[46] G. Folino, C. Pizzuti, and G. Spezzano, "A cellular genetic programming approach to classification," in *Proc. Genet. Evol. Comput. Conf.*, San Mateo, CA: Morgan Kaufmann, Jul. 1999, pp. 1015–1020.

[47] G. Folino, C. Pizzuti, and G. Spezzano, "Genetic programming and simulated annealing: A hybrid method to evolve decission trees," in *Proc. Genet. Program. (EuroGP 2000)* (Lecture Notes in Computer Science Series, 1802). Berlin, Germany: Springer-Verlag, Apr., pp. 294–303.

[48] H. Zhao, "A multi-objective genetic programming approach to developing pareto optimal decision trees," *Decis. Support Syst.*, vol. 43, no. 3, pp. 809–826, Apr. 2007.

[49] J. Eggermont, "Evolving fuzzy decision trees with genetic programming and clustering," in *Proc. 5th Eur. Conf. Genet. Program. (EuroGP 2002)* (Lecture Notes in Computer Science, 2278), Berlin, Germany: Springer-Verlag, Apr., pp. 71–82.

[50] S. Rouwhorst and A. Engelbrecht, "Searching the forest: Using decision trees as building blocks for evolutionary search in classification databases," in *Proc. 2000 Congr. Evol. Comput.*, vol. 1, La Jolla, CA: IEEE, Jul., pp. 633–638.

[51] M. C. J. Bot and W. B. Langdon, "Application of genetic programming to induction of linear classification trees," in *Proc. Genet. Program. (EuroGP)* (Lecture Notes in Computer Science Series), vol. 1802, Berlin, Germany: Springer-Verlag, Apr. 2000, pp. 247–258.

[52] M. C. J. Bot, "Improving induction of linear classification trees with genetic programming," in *Proc. Genet. Evol. Comput. Conf.*, San Mateo, CA: Morgan Kaufmann, 2000, pp. 403–410.

[53] R. E. Marmelstein and G. B. Lamont, "Pattern classification using a hybrid genetic program - decision tree approach," in *Proc. 3rd Annu. Conf. Genet. Program.*, San Mateo, CA: Morgan Kaufmann, Jul. 1998, pp. 223–231.

[54] A. Tsakonas, "A comparison of classification accuracy of four genetic programming-evolved intelligent structures," *Inf. Sci.*, vol. 176, no. 6, pp. 691–724, Mar. 2006.

[55] V. Slavov and N. I. Nokolaev, "Fitness landscapes and inductive genetic programming," in *Proc. 3rd Int. Conf. Artif. Neural Nets Genet. Algorithms*, Berlin, Germany: Springer-Verlag, 1997, pp. 414–418.

[56] J. R. Koza, "Concept formation and decision tree induction using the genetic programming paradigm," in *Proc. 1st Workshop Parallel Probl. Solving Nat.* (Lecture Notes in Computer Science Series, 496). Berlin, Germany: Springer-Verlag, Oct. 1990, pp. 124–128.

[57] C. Elkan, "The foundations of cost-sensitive learning," in *Proc. 17th Int. Joint Conf. Artif. Intell. (IJCAI 2001)*, vol. 2, San Mateo, CA: Morgan Kaufmann, pp. 973–978.

[58] T. Fawcett, "ROC graphs: Notes and practical considerations for data mining researchers," HP, Tech. Rep. HPL-2003-4, 2003.

[59] A. A. Freitas, *Data Mining and Knowledge Discovery With Evolutionary Algorithms*. Berlin, Germany: Springer-Verlag, 2002.

[60] H. E. Johnson, R. J. Gilbert, M. K. Winson, R. Goodacre, A. R. Smith, J. J. Rowland, M. A. Hall, and D. B. Kell, "Explanatory analysis of the metabalone using genetic programming of simple, interpretable rules," *Genet. Program. Evol. Mach.*, vol. 1, no. 3, pp. 243–258, Jul. 2000.

[61] S. X. Wang and P. Lichodzijewski, "Boolean genetic programming for promoter recognition in eukaryotes," in *Proc. IEEE Congr. Evol. Comput.*, Edimburgh, UK: IEEE, Sep. 2005, pp. 683–690.

[62] R. J. Gilbert, J. J. Rowland, and D. B. Kell, "Genomic computing: Explanatory modelling for functional genomics," in *Proc. Genet. Evol. Comput. Conf.*, San Mateo, CA: Morgan Kaufmann, Jul. 2000, pp. 551–557.

[63] J. Yu, J. Yu, A. A. Almal, S. M. Dhanasekaran, D. Ghosh, W. P. Worzel, and A. M. Chinnaiyan, "Feature selection and molecular classification of cancer using genetic programming," *Neoplasia*, vol. 9, no. 4, pp. 292–303, Apr. 2007.

[64] S. Shen, W. Sandham, M. H. Granat, M. F. Dempsey, and J. Patterson, "A new approach to brain tumour diagnosis using fuzzy logic based genetic programming," in *Proc. 25th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, vol. 1, Piscataway, NJ: IEEE, Sep. 2003, pp. 870–873.

[65] K. C. Tan, Q. Yu, C. M. Heng, and T. H. Lee, "Evolutionary computing for knowledge discovery in medical diagnosis," *Artif. Intell. Med.*, vol. 27, no. 2, pp. 129–154, 2003.

[66] C. C. Bojarczuk, H. S. Lopes, A. A. Freitas, and E. L. Michalkiewicz, "A constrained-syntax genetic programming system for discovering classification rules: Application to medical data sets," in *Artif. Intell. Med.*, vol. 30, no. 1, pp. 27–48, 2004, ISSN 0933-3657.

[67] C. C. Bojarczuk, H. S. Lopes, and A. A. Freitas, "Genetic programming for knowledge discovery in chest pain diagnosis," *IEEE Eng. Med. Biol. Mag.*, vol. 19, no. 4, pp. 38–44, Jul./Aug. 2000.

[68] P. S. Ngan, M. L. Wong, W. Lam, K. S. Leung, and J. C. Y. Cheng, "Medical data mining using evolutionary computation," *Artif. Intell. Med.*, vol. 16, no. 1, pp. 73–96, May 1999.

[69] M. L. Wong, W. Lam, K. S. Leung, P. S. Ngan, and J. C. Y. Cheng, "Discovering knowledge from medical databases using evolutionary algorithms," *IEEE Eng. Med. Biol. Mag.*, vol. 19, no. 4, pp. 45–55, Jul./Aug. 2000.

[70] A. Tsakonas, G. Dounias, J. Jantzen, H. Axer, B. Bjerregaard, and D. G. v. Keyserlingk, "Evolving rule-based systems in two medical domains using genetic programming," *Artif. Intell. Med.*, vol. 32, no. 3, pp. 195–216, Nov. 2004.

[71] C. Qing-Shan, Z. De Fu, W. Li-Jun, and C. Huo-Wang, "A modified genetic programming for behavior scoring problem," in *Proc. IEEE Symp. Comput. Intell. Data Mining*. Honolulu, Hawaii: IEEE, Apr. 2007, pp. 535–539.

[72] S. Sakprasat and M. C. Sinclair, "Classification rule mining for automatic credit approval using genetic programming," in *Proc. IEEE Congr. Evol. Comput.*, Singapore: IEEE, Sep. 2007, pp. 548–555.

[73] P. J. Bentley, "'Evolutionary, my dear Watson': Investigating committee-based evolution of fuzzy rules for the detection of suspicious insurance claims," in *Proc. Genet. Evol. Comput. Conf.*, San Mateo, CA: Morgan Kaufmann, Jul. 2000, pp. 702–709.

[74] A. L. Garcia-Almanza and E. P. K. Tsang, "Evolving decision rules to predict investment opportunities," *Int. J. Autom. Comput.*, vol. 5, no. 1, pp. 22–31, Jan. 2008.

[75] S. A. Stanhope and J. M. Daida, "Genetic programming for automatic target classification and recognition in synthetic aperture radar imagery," in *Proc. 7th Annu. Conf. Evol. Program. VII* (Lecture Notes in Computer Science Series, 1447). Mission Valley Marriott, San Diego, CA: Springer-Verlag, 25–27, 1998, pp. 735–744.

[76] C. D. Stefano, A. D. Cioppa, and A. Marcelli, "Character preclassification based on genetic programming," *Pattern Recogn. Lett.*, vol. 23, no. 12, pp. 1439–1448, Oct. 2002.

[77] I. De Falco, A. Della Cioppa, and E. Tarantino, "Discovering interesting classification rules with genetic programming," *Appl. Soft Comput. J.*, vol. 1, no. 4, pp. 257–269, 2002.

[78] K. C. Tan, A. Tay, T. H. Lee, and C. M. Heng, "Mining multiple comprehensible classification rules using genetic programming," in *Proc. 2002 Congr. Evol. Comput.*, vol. 2, Piscataway, NJ: IEEE, May, pp. 1302–1307.

[79] E. C. no, G. Leguizamón, and N. Wagner, "Evolution of classification rules for comprehensible knowledge discovery," in *Proc. IEEE Congr. Evol. Comput.*, Singapore: IEEE, Sep. 2007, pp. 1261–1268.

[80] M. L. Wong and K. S. Leung, *Data Mining using Grammar-Based Genetic Programming and Applications*. Norwell, MA: Kluwer, 2000.

[81] F. J. Berlanga, M. J. del Jesus, and F. Herrera, "A novel genetic cooperative-competitive fuzzy rule based learning method using genetic programming for high dimensional problems," in *Proc. 3rd Int. Workshop Genet. Evol. Fuzzy Syst.*. Witten-Bommerholz, Germany: IEEE, Mar. 2008, pp. 101–106.

[82] J. Eggermont, A. E. Eiben, and J. I. van Hemert, "Adapting the fitness function in GP for data mining," in *Proc. 2nd Eur. Workshop, Genet. Program. (EuroGP)* (ser. Lecture Notes in Computer Science, 1598). Berlin, Germany: Springer-Verlag, May 1999, pp. 193–202.

[83] J. Eggermont, A. E. Eiben, and J. I. van Hemert, "A comparison of genetic programming variants for data classification," in *Proc. 3rd Int. Symp. Adv. Intell. Data Anal. (IDA)* (Lecture Notes in Computer Science, 1642). Berlin, Germany: Springer-Verlag, Aug. 1999, pp. 281–290.

[84] S. García, F. González, and L. Sánchez, "Evolving fuzzy rule based classifiers with GA-P: A grammatical approach," in *Proc. 2nd Eur. Workshop, Genet. Program. (EuroGP)* (Lecture Notes in Computer Science Series, 1598). Berlin, Germany: Springer-Verlag, May 1999, pp. 203–210.

[85] R. R. F. Mendes, F. de B. Voznika, J. C. Nievola, and A. A. Freitas, "Discovering fuzzy classification rules with genetic programming and co-evolution," in *Proc. Genet. Evol. Comput. Conf.*, San Mateo, CA: Morgan Kaufmann, Jul. 2001, p. 183.

[86] P. G. Espejo, C. Romero, S. Ventura, and C. Hervás, "Induction of classification rules with grammar-based genetic programming," in *Proc. 2nd Int. Conf. Mach. Intell. (ACIDCA ICMI)*, Tozeur, Tunisia, Nov. 2005, pp. 596–601.

[87] R. Cattral, F. Oppacher, and D. Deugo, "Supervised and unsupervised data mining with an evolutionary algorithm," in *Proc. 2001 Congr. Evol. Comput.*, vol. 2, Piscataway, NJ: IEEE, May, pp. 767–774.

[88] L. M. Howard and D. J. D'Angelo, "The GA-P: A genetic algorithm and genetic programming hybrid," *IEEE Exp.*, vol. 10, no. 3, pp. 11–15, Jun. 1995.

[89] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies - A comprehensive introduction," *Nat. Comput.*, vol. 1, no. 1, pp. 3–52, Mar. 2002.

[90] D. Lennartsson and P. Nordin, "A genetic programming method for the identification of signal peptides and prediction of their cleavage sites," *EURASIP J. Appl. Signal Process.*, vol. 2004, no. 1, pp. 138–145, Jan. 2004.

[91] H. F. Gray, R. J. Maxwell, I. Martínez-Pérez, C. Arús, and S. Cerdán, "Genetic programming for classification of brain tumours from nuclear magnetic resonance biopsy spectra," in *Proc. 1st Annu. Conf. Genet. Program.*, Stanford, CA: MIT Press, Jul. 1996, p. 424..

[92] D. Hope, E. Munday, and S. Smith, "Evolutionary algorithms in the classification of mammograms," in *Proc. IEEE Symp. Comput. Intell. Image Signal Process.*, Piscataway, NJ: IEEE, Apr. 2007, pp. 258–265.

[93] W. Wongseree, N. Chaiyaratana, K. Vichittumaros, P. Winichagoon, and S. Fucharoen, "Thalassaemia classification by neural networks and genetic programming," *Inf. Sci.*, vol. 177, no. 3, pp. 771–786, Feb. 2007.

[94] T. K. Paul and H. Iba, "Classification of scleroderma and normal biopsy data and identification of possible biomarkers of the disease," in *Proc. IEEE Symp. Comput. Intell. Bioinf. Comput. Biol.*, Toronto, ON: IEEE, Sep. 2006, pp. 306–311.

[95] G. C. Wilson and M. I. Heywood, "Introducing probabilistic adaptive mapping developmental genetic programming with redundant mappings," *Genet. Program. Evol. Mach.*, vol. 8, no. 2, pp. 187–220, Jun. 2007.

[96] J.-Y. Lin, H.-R. Ke, B.-C. Chien, and W.-P. Yang, "Classifier design with feature selection and feature extraction using layered genetic programming," *Exp. Syst. Appl.*, vol. 34, no. 2, pp. 1384–1393, Feb. 2008.

[97] S. Winkler, M. Affenzeller, and S. Wagner, "Advanced genetic programming based machine learning," *J. Math. Model. Algorithms*, vol. 6, no. 3, pp. 455–480, 2007.

[98] M. Zhang and P. Wong, "Genetic programming for medical classification: A program simplification approach," *Genet. Program. Evol. Mach.*, vol. 9, pp. 229–255, 2008.

[99] M. Brameier and W. Banzhaf, "A comparison of linear genetic programming and neural networks in medical data mining," *IEEE Trans. Evol. Comput.*, vol. 5, no. 1, pp. 17–26, Feb. 2001.

[100] S. Silva and Y.-T. Tseng, "Classification of seafloor habitats using genetic programming," in *Proc. Appl. Evol. Comput. (EvoWorkshops)*. (Lecture Notes in Computer Science, 4974). Naples, Italy: Springer-Verlag, Mar. 2008, pp. 315–324.

[101] A. Vieira, B. Ribeiro, S. Mukkamala, J. C. Neves, and A. H. Sung, "On the performance of learning machines for bankruptcy detection," in *Proc. 2nd IEEE Int. Conf. Comput. Cybern.*, Piscataway, NJ: IEEE, 2004, pp. 323–327.

[102] T. Lensberg, A. Eilifsen, and T. E. McKee, "Bankruptcy theory development and classification via genetic programming," *Eur. J. Oper. Res.*, vol. 169, no. 2, pp. 677–697, Mar. 2006.

[103] S. Sette, B. Wyns, and L. Boullart, "Comparing learning classifier systems and genetic programming: A case study," *Eng. Appl. Artif. Intell.*, vol. 17, no. 2, pp. 199–204, Mar. 2004.

[104] L. Zhang and A. K. Nandi, "Fault classification using genetic programming," *Mech. Syst. Signal Process.*, vol. 21, no. 3, pp. 1273–1284, Apr. 2007.

[105] L. Zhang, L. B. Jack, and A. K. Nandi, "Fault detection using genetic programming," *Mech. Syst. Signal Process.*, vol. 19, no. 2, pp. 271–289, Mar. 2005.

[106] K. Hennessy, M. G. Madden, J. Conroy, and A. G. Ryder, "An improved genetic programming technique for the classification of Raman spectra," *Knowl. Based Syst.*, vol. 18, no. 4–5, pp. 217–224, Aug. 2005.

[107] Y. Zhang, H. Li, M. Niranjan, and P. Rockett, "Applying cost-sensitive multiobjective genetic programming to feature extraction for spam e-mail filtering," in *Proc. 11th Eur. Conf. Genet. Program.* (Lecture Notes in Computer Science Series, 4971). Naples, Italy: Springer-Verlag, Mar. 2008, pp. 325–336.

[108] K. Faraoun and A. Boukelif, "Genetic programming approach for multicategory pattern classification applied to network intrusions detection," *Int. Arab J. Inf. Technol.*, vol. 4, no. 3, pp. 237–246, Jul. 2007.

[109] S. Mukkamala, A. H. Sung, and A. Abraham, "Modeling intrusion detection systems using linear genetic programming approach," in *Proc. 17th Int. Conf. Ind. Eng. Appl. Artif. Intell. Exp. Syst. - Innovations Appl. Artif. Intell.* (Lecture Notes in Computer Science Series, 3029). Ottawa, Canada: Springer-Verlag, May 2004, pp. 633–642.

[110] Y. Liu, T. Khoshgoftaar, and J.-F. Yao, "Building a novel GP-based software quality classifier using multiple validation datasets," in *Proc. IEEE Int. Conf. Inf. Reuse Integr.*, Las Vegas, NV: IEEE, Aug. 2007, pp. 644–650.

[111] W. A. Tackett, "Genetic programming for feature discovery and image discrimination," in *Proc. 5th Int. Conf. Genet. Algorithms*, San Mateo, CA: Morgan Kaufmann, Jul. 1993, pp. 303–309.

[112] A. M. Teredesai and V. Govindaraju, "Issues in evolving GP based classifiers for a pattern recognition task," in *Proc. IEEE Congr. Evol. Comput.*, vol. 1, Portland, Oregon: IEEE, Jun. 2004, pp. 509–515.

[113] G. Wijesinghe and V. Ciesielski, "Using restricted loops in genetic programming for image classification," in *Proc. IEEE Congr. Evol. Comput.*, Singapore: IEEE, Sep. 2007, pp. 4569–4576.

[114] P. J. Rauss, J. M. Daida, and S. A. Chaudhary, "Classification of spectral image using genetic programming," in *Proc. Genet. Evol. Comput. Conf.*, Las Vegas, Nevada: Morgan Kaufmann, Jul. 2000, pp. 726–733.

[115] N. Petrović and V. S. Crnojević, "Impulse noise detection based on robust statistics and genetic programming," in *Proc. 7th Int. Conf. Adv. Concepts Intell. Vis. Syst.* (Lecture Notes in Computer Science Series, 3708), Antwerp, Belgium: Springer-Verlag, Sep. 2005, pp. 643–649.

[116] D. Agnelli, A. Bollini, and L. Lombardi, "Image classification: An evolutionary approach," *Pattern Recogn. Lett.*, vol. 23, no. 1–3, pp. 303–309, Jan. 2002.

[117] A. Song and V. Ciesielski, "Texture analysis by genetic programming," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2, Portland, OR: IEEE, Jun. 2004, pp. 2092–2099.
[118] Z. Chen and S. Lu, "A genetic programming approach for classification of textures based on wavelet analysis," in *Proc. Int. Symp. Intell. Signal Process.*, Piscataway, NJ: IEEE, Oct. 2007, pp. 1–6.
[119] M. Zhang and W. D. Smart, "Multiclass object classification using genetic programming," in *Proc. Appl. Evol. Comput. EvoWorkshops 2004*, (Lecture Notes in Computer Science Series, 3005). Coimbra, Portugal: Springer-Verlag, Apr., pp. 369–378.
[120] Y.-M. Li, M. Wang, L.-J. Cui, and D.-M. Huang, "A new classification arithmetic for multi-image classification in genetic programming," in *Proc. 6th Int. Conf. Mach. Learning Cybern.*, vol. 3, Hong Kong: IEEE, Aug. 2007, pp. 1683–1687.
[121] M. Zhang and W. D. Smart, "Using Gaussian distribution to construct fitness functions in genetic programming for multiclass object classification," *Pattern Recogn. Lett.*, vol. 27, no. 11, pp. 1266–1274, Aug. 2006.
[122] M. Zhang, X. Gao, and W. Lou, "A new crossover operator in genetic programming for object classification," *IEEE Trans. Syst., Man, Cybern. B*, vol. 37, no. 5, pp. 1332–1343, Oct. 2007.
[123] A. Teller and M. Veloso, "Program evolution for data mining," *Int. J. Exp. Syst.*, vol. 8, no. 3, pp. 213–236, 1995.
[124] G. Patterson and M. Zhang, "Fitness functions in genetic programming for classification with unbalanced data," in *Proc. 20th Australian Joint Conf. Artif. Intell. - Adv. Artif. Intell.* (Lecture Notes in Computer Science Series), vol. 4830. Gold Coast, Australia: Springer-Verlag, Dec. 2007, pp. 769–775.
[125] R. Curry, P. Lichodzijewski, and M. I. Heywood, "Scaling genetic programming to large datasets using hierarchical dynamic subset selection," *IEEE Trans. Syst., Man, Cybern. B*, vol. 37, no. 4, pp. 1065–1073, Aug. 2007.
[126] G. Li, J. F. Wang, K. H. Lee, and K.-S. Leung, "Instruction-matrix-based genetic programming," *IEEE Trans. Syst., Man, Cybern. B*, vol. 38, no. 4, pp. 1036–1049, Aug. 2008.
[127] M. J. Cavaretta and K. Chellapilla, "Data mining using genetic programming - the implications of parsimony on generalization error," in *Proc. Congr. Evol. Comput.*, vol. 2, Piscataway, NJ: IEEE, Jul. 1999, pp. 1330–1337.
[128] J.-Y. Lin, H.-R. Ke, B.-C. Chien, and W.-P. Yang, "Designing a classifier by a layered multi-population genetic programming approach," *Pattern Recogn.*, vol. 40, no. 8, pp. 2211–2225, Aug. 2007.
[129] D. P. Muni, N. R. Pal, and J. Das, "A novel approach to design classifiers using genetic programming," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 183–196, Apr. 2004.
[130] M. Lemczyk and M. I. Heywood, "Training binary GP classifiers efficiently: A pareto-coevolutionary approach," in *Proc. 10th Eur. Conf. Genet. Program.* (Lecture Notes in Computer Science Series, 4445), Berlin, Germany: Springer-Verlag, Apr. 2007, pp. 229–240.
[131] J. K. Kishore, L. M. Patnaik, V. Mani, and V. K. Agrawal, "Application of genetic programming for multicategory pattern classification," *IEEE Trans. Evol. Comput.*, vol. 4, no. 3, pp. 242–258, Sep. 2000.
[132] T. Loveard and V. Ciesielski, "Representing classification problems in genetic programming," in *Proc. IEEE Congr. Evol. Comput.*. vol. 2, Seoul, South Korea: IEEE, May 2001, pp. 1070–1077.
[133] M. Oltean and C. G. san, "Solving classification problems using infix form genetic programming," in *Proc. 5th Int. Symp. Intell. Data Anal. - Adv. Intell. Data Anal. V*, (ser. Lecture Notes in Computer Science), vol. 2810. Berlin, Germany: Springer-Verlag, Aug. 2003, pp. 242–253.
[134] B.-C. Chien, J. Y. Lin, and T.-P. Hong, "Learning discriminant functions with fuzzy attributes for classification using genetic programming," *Exp. Syst. Appl.*, vol. 23, no. 1, pp. 31–37, Jul. 2002.
[135] B.-C. Chien and J.-H. Yang, "Features selection based on rough membership and genetic programming," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, 2006, vol. 5, pp. 4124–4129.
[136] B.-C. Chien, J.-H. Yang, and W.-Y. Lin, "Generating effective classifiers with supervised learning of genetic programming," in *Proc. 5th Int. Conf. Data Warehousing Knowl. Discov.*, (Lecture Notes in Computer Science Series, 2737). Prague, Czech Republic: Springer-Verlag, Sep. 2003, pp. 192–201.
[137] B.-C. Chien, J.-Y. Lin, and W.-P. Yang, "Learning effective classifiers with Z-value measure based on genetic programming," *Pattern Recogn.*, vol. 37, no. 10, pp. 1957–1972, Oct. 2004.
[138] M. D. Ritchie, A. A. Motsinger, W. S. Bush, C. S. Coffey, and J. H. Moore, "Genetic programming neural networks: A powerful bioinformatics tool for human genetics," *Appl. Soft Comput.*, vol. 7, no. 1, pp. 471–479, Jan. 2007.
[139] R. K. Rao, S. Lakshminarayanan, and K. Tun, "Genetic programming models for classification of data from biological systems," in *Proc. IEEE Congr. Evol. Comput.*, Singapore: IEEE, Sep. 2007, pp. 4154–4161.
[140] J.-J. Huang, G.-H. Tzeng, and C.-S. Ong, "Two-stage genetic programming (2SGP) for the credit scoring model," *Appl. Math. Comput.*, vol. 174, no. 2, pp. 1039–1053, Mar. 2006.
[141] A. Tsakonas, G. Dounias, M. Doumpos, and C. Zopounidis, "Bankruptcy prediction with neural logic networks by means of grammar-guided genetic programming," *Exp. Syst. Appl.*, vol. 30, no. 3, pp. 449–461, Apr. 2006.
[142] C. G. Doherty, "Fundamental analysis using genetic programming for classification rule induction," in *Proc. Genet. Algorithms Genet. Program. Stanford 2003*, Stanford, CA: Stanford Bookstore, pp. 45–51, [Online]. Available: http://www.genetic-programming.org/sp2003/Doherty.pdf
[143] L. Hirsch, R. Hirsch, and M. Saeedi, "Evolving Lucene search queries for text classification," in *Proc. 2007 Genet. Evol. Comput. Conf.*, London, England: ACM, Jul., pp. 1604–1611.
[144] A. Majid, A. Khan, and A. M. Mirza, "Improving performance of nearest neighborhood classifier using genetic programming," in *Proc. IEEE Int. Conf. Mach. Learning Appl.*, Louisville, Kentucky: IEEE, Dec. 2004, pp. 469–476.
[145] L. Diosan, A. Rogozan, and J.-P. Pécuchet, "Optimising multiple kernels for SVM by genetic programming," in *Proc. 8th Eur. Conf. Evol. Comput. Comb. Optim.*, (Lecture Notes in Computer Science Series 4972). Naples, Italy: Springer-Verlag, Mar. 2008, pp. 230–241.
[146] T. Howley and M. G. Madden, "The genetic kernel support vector machine: Description and evaluation," *Artif. Intell. Rev.*, vol. 24, no. 3–4, pp. 379–395, Nov. 2005.
[147] K. Sullivan and S. Luke, "Evolving kernels for support vector machine classification," in *Proc. 2007 Genet. Evol. Comput. Conf.*, London, England: ACM, Jul., pp. 1702–1707.
[148] T. Phienthrakul and B. Kijsirikul, "GPES: An algorithm for evolving hybrid kernel functions of support vector machines," in *Proc. IEEE Congr. Evol. Comput.*, Singapore: IEEE, Sep. 2007, pp. 2636–2643.
[149] M. Gîrdea and L. Ciortuz, "A hybrid genetic programming and boosting technique for learning kernel functions from training data," in *Proc. 9th Int. Symp. Symbolic Numeric Algorithms Sci. Comput.*, Timisoara, Romania: IEEE, Sep. 2007, pp. 395–402.
[150] T. Watson and T. Rakowski, "Data mining with an evolving population of database queries," in *Proc. MENDEL 1995*, pp. 169–174.
[151] C. Gagné, M. Schoenauer, M. Sebag, and M. Tomassini, "Genetic programming for kernel-based learning with co-evolving subsets selection," in *Proc. 9th Int. Conf. Parallel Probl. Solving Nat.* (Lecture Notes in Computer Science, 4193), Reykjavik, Iceland: Springer-Verlag, Sep. 2006, pp. 1008–1017.
[152] P. Lichodzijewski and M. I. Heywood, "GP classifier problem decomposition using first-price and second-price auctions," in *Proc. 10th Eur. Conf. Genet. Program.* (Lecture Notes in Computer Science Series, 4445), Valencia, Spain: Springer-Verlag, Apr. 2007, pp. 137–147.
[153] P. Lichodzijewski and M. I. Heywood, "Coevolutionary bid-based genetic programming for problem decomposition in classification," *Genet. Program. Evol. Mach.*, vol. 9, no. 4, pp. 331–365, Dec. 2008.
[154] C. Gagné and M. Parizeau, "Coevolution of nearest neighbor classifiers," *Int. J. Pattern Recogn. Artif. Intell.*, vol. 21, no. 5, pp. 921–946, 2007.
[155] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, Jul. 1998.
[156] V. N. Vapnik, *The Nature of Statistical Learning*, 2nd ed. Berlin, Germany: Springer-Verlag, 2000.
[157] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge, U.K.: Cambridge Univ. Press, 2000.
[158] A. A. Freitas, "A genetic programming framework for two data mining tasks: Classification and generalized rule induction," in *Proc. 2nd Annu. Conf. Genet. Program.*, San Mateo, CA: Morgan Kaufmann, 1997, pp. 96–101.
[159] C. Y. Ishida and A. T. R. Pozo, "GPSQL Miner: SQL-grammar genetic programming in data mining," in *Proc. Congr. Evol. Comput.*, vol. 2, Piscataway, NJ: IEEE, May 2002, pp. 1226–1231.
[160] K. Yu, L. Ji, and X. Zhang, "Kernel nearest neighbor algorithm," *Neural Process. Lett.*, vol. 15, no. 2, pp. 147–156, 2002.
[161] E. B. Baum and I. Durdanovic, "An artificial economy of post production systems," in *Proc. 3rd Int. Workshop Adv. Learning Classifier*

Syst., (Lecture Notes in Computer Science Series, 1996). Paris, France: Springer-Verlag, Sep. 2000, pp. 3–20.
[162] E. Baum and I. Durdanovic, "Toward code evolution by artificial economies," in *Evolution as Computation*, L. F. Landweber and E. Winfree, Eds. Berlin, Germany: Springer-Verlag, 2002, pp. 314–332.
[163] L. Breiman, "Bagging predictors," *Mach. Learning*, vol. 24, no. 2, pp. 123–140, Aug. 1996.
[164] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Mach. Learning*, vol. 36, no. 1–2, pp. 105–139, Jul. 1999.
[165] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization," *Mach. Learning*, vol. 40, no. 2, pp. 139–157, Aug. 2000.
[166] M. Brameier and C. Wiuf, "Ab initio identification of human microRNAs based on structure motifs," *BMC Bioinf.*, vol. 8, p. 478, Dec. 2007.
[167] C.-G. Xu and K.-H. Liu, "A GP based approach to the classification of multiclass microarray datasets," in *Proc. 4th Int. Conf. Intell. Comput.*, (Lecture Notes in Computer Science Series, 5227). Shanghai, China: Springer-Verlag, Sep. 2008, pp. 340–346.
[168] K. Imamura, T. Soule, R. B. Heckendorn, and J. A. Foster, "Behavioral diversity and a probabilistically optimal GP ensemble," *Genet. Program. Evol. Mach.*, vol. 4, no. 3, pp. 235–253, Sep. 2003.
[169] J.-H. Hong and S.-B. Cho, "The classification of cancer based on DNA microarray data that uses diverse ensemble genetic programming," *Artif. Intell. Med.*, vol. 36, no. 1, pp. 43–58, Jan. 2006.
[170] S. Hengpraprohm and P. Chongstitvatana, "A genetic programming ensemble approach to cancer microarray data classification," in *Proc. 3rd Int. Conf. Innovative Comput. Inf. Control*, Piscataway, NJ: IEEE, Jun. 2008, pp. 340–340.
[171] M. Brameier and W. Banzhaf, "Evolving teams of predictors with linear genetic programming," *Genet. Program. Evol. Mach.*, vol. 2, no. 4, pp. 381–407, Dec. 2001.
[172] W. B. Langdon, S. J. Barret, and B. F. Buxton, "Combining decision trees and neural networks for drug discovery," in *Proc. 5th Eur. Conf., Genet. Program. (EuroGP)*. (Lecture Notes in Computer Science, 2278). Berlin, Germany: Springer-Verlag, Apr. 2002, pp. 60–70.
[173] Y. Zhang and S. Bhattacharyya, "Genetic programming in classifying large-scale data: An ensemble method," *Inf. Sci.*, vol. 163, no. 1–3, pp. 85–101, Jun. 2004.
[174] G. Folino, C. Pizzuti, and G. Spezzano, "Training distributed GP ensemble with a selective algorithm based on clustering and pruning for pattern classification," *IEEE Trans. Evol. Comput.*, vol. 12, no. 4, pp. 458–468, Aug. 2008.
[175] R. Thomason and T. Soule, "Novel ways of improving cooperation and performance in ensemble classifiers," in *Proc. Genet. Evol. Comput. Conf.*, London, England: ACM, Jul. 2007, pp. 1708–1715.
[176] G. Folino, C. Pizzuti, and G. Spezzano, "Mining distributed evolving data streams using fractal GP ensembles," in *Proc. 10th Eur. Conf. Genet. Program.*, (Lecture Notes in Computer Science Series, 4445). Valencia, Spain: Springer-Verlag, Apr. 2007, pp. 160–169.
[177] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice-Hall, 1988.
[178] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural Comput.*, vol. 8, no. 7, pp. 1341–1390, Oct. 1996.
[179] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.
[180] P. Whigham, "Gramatical bias for evolutionary learning," Ph.D. dissertation, School of Comput. Sci.; Univ. College; Univ. of New South Wales; Australian Defence Force Academy, 1996.
[181] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Editorial: Special issue on learning from imbalanced data sets," *SIGKDD Explorations*, vol. 6, no. 1, pp. 1–6, Jun. 2004.
[182] G. M. Weiss, "Mining with rarity: A unifying framework," *SIGKDD Explorations*, vol. 6, no. 1, pp. 7–19, Jun. 2004.

Pedro G. Espejo was born in Cordoba, Spain, in 1973. He received the B.Sc. degree from the University of Cordoba, Cordoba, in 1994, and the M.Sc. degree from the University of Granada, Granada, Spain, in 1996, both in computer science.

He was an Assistant Teacher with the Computing Languages and Systems Department, University of Cadiz, Cadiz, Spain, from 1997 to 2003, and a Counselor Teacher with the National University of Distance Education, Jerez de la Frontera, Cadiz, from 2000 to 2003. Since 2003, he has been an Assistant Teacher with the Department of Computer Science and Numerical Analysis, University of Cordoba. His research interests include machine learning, data mining, genetic programming, and learning from unbalanced data.

Sebastián Ventura was born in Cordoba, Spain, in 1966. He received the B.Sc. and Ph.D. degrees in sciences from the University of Cordoba, Cordoba, in 1989 and 1996, respectively.

He is currently an Associate Professor with the Department of Computer Science and Numerical Analysis, University of Cordoba, where he heads the Knowledge Discovery and Intelligent Systems Research Laboratory. He is the author or coauthor of more than 60 international publications, 20 of them published in international journals. He has also been engaged on 11 research projects (being the coordinator of two of them) supported by the Spanish and Andalusian governments and the European Union, concerning several aspects in the area of evolutionary computation, machine learning, data mining, and their applications. His current main research interests are in the fields of soft-computing, machine learning, data mining and its applications.

Dr. Ventura is Member or the IEEE Computer, Computational Intelligence and Systems, Man and Cybernetics societies and the Association of Computing Machinery.

Francisco Herrera received the M.Sc. and Ph.D. degrees in mathematics from the University of Granada, Granada, Spain, in 1988 and 1991, respectively.

He is currently a Professor with the Department of Computer Science and Artificial Intelligence, University of Granada. He is the author or coauthor of more than 150 papers in international journals. He is coauthor of the book Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases (Singapore: World Scientific, 2001). He has coedited five international books and coedited twenty special issues in international journals on different soft computing topics. His current research interests include computing with words and decision making, data mining, data preparation, instance selection, fuzzy rule based systems, genetic fuzzy systems, knowledge extraction based on evolutionary algorithms, memetic algorithms, and genetic algorithms.

Dr. Herrera is an Associated Editor of the following journals: IEEE Transactions on Fuzzy Systems, Mathware and Soft Computing, Advances in Fuzzy Systems, Advances in Computational Sciences and Technology, and International Journal of Applied Metaheuristics Computing. He currently serves as Area Editor of the Journal Soft Computing (area of genetic algorithms and genetic fuzzy systems), and also serves as a Member of Editorial Boards of several journals, which includes: Fuzzy Sets and Systems, Applied Intelligence, Knowledge and Information Systems, Information Fusion, Evolutionary Intelligence, International Journal of Hybrid Intelligent Systems, Memetic Computation.