Advanced Review

Evolutionary design of decision trees for medical application

Peter Kokol,[1]* Sandi Pohorec,[2] Gregor Štiglic[1] and Vili Podgorelec[2]

Decision trees (DT) are a type of data classifiers. A typical classifier works in two phases. In the first, the learning phase, the classifier is built according to a preexisting data (training) set. Because decision trees are being induced from a known training set, and the labels on each example are known the first step can also be referred to as supervised learning. The second step is when the induced classifier is used for classification. Usually, prior to the first step several steps should be performed to improve the accuracy and efficiency of the classification: data cleaning, redundancy elimination, and data normalization. Classifiers are evaluated for accuracy, speed, robustness, scalability, and interpretability. DTs are widely used for exploratory knowledge discovery where comprehensible knowledge representation is preferred. The main attraction of DTs lies in the intuitive representation that is easy to understand and comprehend. Accuracy, however, is dependent on the learning data. One of the methods to improve the induction and other phases in the creation of a classifier is the use of evolutionary algorithms. They are used because the classic deterministic approach is not necessarily optimal with regard to the quality, accuracy, and complexity of the obtained classifier. In addition to the description of different evolutionary DT induction approaches, this paper also presents multiple examples of evolutionary DT applications in the medical domain.

How to cite this article:
WIREs Data Mining Knowl Discov 2012. doi: 10.1002/widm.1056

INTRODUCTION

Practical decision making in medicine can be particularly problematic, involving a variety of complex diagnostics and therapeutic uncertainty, cost, preferences, values and similar, in addition to the fact that they have substantial consequences including death of a patient. When facing such complex decisions, it can be difficult to comprehend and compare all various options and alternatives just "in our heads." We need to have aids to help us gain insight into the variables, features, and attributes that might have a major impact on our decisions.

This paper focuses on evolutionary decision trees (EDTs) and how they can be applied to medical problems. Most of the novel evolutionary decision tree building algorithms are tested on a wide variety of data sets from different fields with data sets from UCI repository being most widely used. In most cases, medical data sets deposited in UCI repository are also being used for the evaluation of novel methods. However, there are also some studies where authors have provided their own data sets from the medical field and evaluated their algorithms on very specific data sets. In most cases, one will meet the following unique features of medical data:

- Missing values as a consequence of accidentally not entered data, or purposely not obtained data for technical or other reasons. Most of modern decision tree building techniques, including evolutionary ones, are able to build classification models on data sets containing missing values. Usually, the problem of missing values is solved by splitting the sample with a missing value to all branches

*Correspondence to: kokol@uni-mb.si
[1]Faculty of Health Sciences, University of Maribor, Maribor, Slovenia
[2]Faculty of Electrical Engineering and Computer Science, University of Maribor, Maribor, Slovenia

DOI: 10.1002/widm.1056

from the node testing the attribute where the value is missing. Alternatively, one can use one of many general techniques for replacing missing data like.[1,2]

- Incorrectness of data originating from systematic or random noise in data sets. Therefore, it is important for a classification model to be made less sensitive to noise. It is also of high importance that training set, test set, and future data set should contain approximately the same level of noise. In cases with relatively few noisy samples, it was demonstrated that pruning of the decision tree creates more general models that perform better in noisy environments compared to more specific decision trees. As described in the following sections of this paper, evolutionary decision trees usually produce less complex and therefore more general models than classical decision tree algorithms.
- Inconsistency of data with typical examples of the same samples categorized as belonging to more than one mutually exclusive class. Only consultation with the staff who were collecting the data or exclusion of such samples can effectively solve the inconsistency of data. A form of inconsistency in data is often associated with incomprehensible or weak description of attributes in a medical data set. Therefore, effective communication with the domain experts (usually medical doctors) is of high importance to obtain useful results.
- Cost-sensitivity is very common in medical diagnosis domain where classification is used to determine the most appropriate procedures for specific patients. The cost of procedures to be used can result in monetary (more vs. less expensive) or quality of life (e.g., invasive vs. noninvasive) consequences. The term cost-based classification includes the cost of tests, the cost of samples, and the misclassification cost.[3–6]

EVOLUTIONARY ALGORITHMS

Evolutionary algorithms (EA) are a family of algorithms that evolve an initial (usually random) population with the use of genetic operators and selection policies. EAs were introduced in the 1950s and 1960s when number of scientists began to study algorithms inspired by natural evolution. Major representatives of evolutionary algorithms are genetic algorithms (GA),[7,8] genetic programming (GP),[8] evolutionary programming (EP),[9] and evolution strategies (ES).[10] GA and GP are very similar and sometimes hard to distinguish. Authors in Ref 11 offer a definition that distinguishes between GA and GP on the basis of whether a solution encodes only data (in that case they classify it as GA) or data and functions (GP). Evolutionary programming and early versions of evolution strategies only use mutation as the source of variation in the population. Evolutionary approaches to design of decision trees are mainly based on GA and GP.

Algorithm 1 Genetic algorithm

Input: *maxgenerations*

1: *numgenerations* ← 0
2: *currentgeneration* ← *GeneratePopulation*()
3: **repeat**
4: Selection
5: Crossover
6: Mutation
7: EvaluateFitness
8: *numgenerations* ←numgenerations + 1
9: **until** *numgenerations* <maxgeneration

FIGURE 1 | Genetic algorithm.

Genetic Algorithms

Genetic algorithm can be defined as a learning and searching algorithm. The standard implementation of a GA is as follows: Initial population (usually generated randomly) goes through a repetition of generations. Every generation is subjected to genetic operators (crossover, mutation) and selection of individuals that advance to the next generation. The selection is based on a fitness evaluation of individuals in the current generation. The repetition is terminated when a stopping condition is reached. The condition can either be a fixed number of generations or a satisfactory solution. The general algorithm for both GA and GP is presented in Figure 1.

Representation

The individuals in the population represent candidate solutions to the given problem. With regard to the problem, a proper candidate representation should be selected. Common types of representations are binary representations, integer representations, real-number representations, and string representations. Also, the representation can be either a fixed length (all chromosomes have the same number of genes) or variable length (some chromosomes are longer than others). It has been shown[12–14] that candidate representations

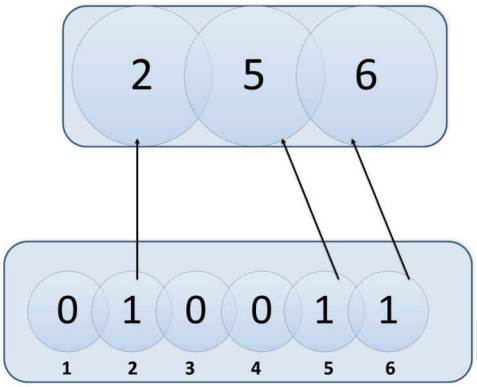

FIGURE 2 | Binary and integer representations of a chromosome.

influence the success of the genetic algorithm. For instance, if a GA is used to select a subset of the training data to improve the classification results, the representation used is either a binary (each chromosome has a gene for each member of the training set, value 1 indicates that the individual is included in the subset, 0 that it is not) or an array of integers (variable length chromosomes contain only the indexes of the training data examples included in the final subset). These representations are shown in Figure 2. The example shows a training set of six individuals of which three (index 2,5,6) are included in the optimized training subset, the binary representation indicates the included individuals with value 1, whereas the integer representation contains only the included individuals and marks them with their index numbers.

Genetic Operators and Selection Types

Crossover operator is used to exchange genetic material between two chromosomes. In evolutionary algorithms, this operator is invoked according to the probability of crossover, which is usually set low (depends on the problem, usually fewer than 20%). It imitates sexual reproduction in nature and produces new chromosomes (offsprings). Basic crossover implementations work by selecting two "parents" from the current generation, then randomly select a position within the parents and exchange the genes. This is called a one-point crossover; other well-known types include a two-point crossover (a substring with start and end point is selected within each parent) and a uniform crossover (parents exchange information at the gene level with multiple point crossover).

The *mutation* operator is used for reclaiming diversity in the population, and it is an instrument of innovation and variation.[7] Mutation changes individual genes in the chromosome and can reanimate lost information from previous generations. Mutation differs from crossover in that it is used for local search: It can only change some properties of an individual and cannot combine properties from multiple individuals.

Selection is a method of choosing individuals for reproduction. Selection promotes fitter individuals to the next generation with the intention that their offspring will be even fitter. Selection should be balanced with other operators: if the selection pressure is too high the diversity of the population will be reduced (the solution converges to a local optimum), if the pressure is too low the evolution will be very slow (many generations with little progress).

Common implementations of the selection process include[15] fitness proportionate selection, tournament selection, and rank-based selection. The *roulette* wheel is an example of a fitness proportionate selection: Each individual has a slice on the wheel, the size is proportional to the individual's fitness, the wheel is spun as many times as there are individuals in the population, and on each spin the individual under the marker is selected to the next generation. Better individuals have larger slices and are therefore more likely to be included in the next generation. The *tournament* selection is done as follows: The tournament size is determined, and the process of selection is repeated as long as there are slots free in the next generation. On each turn, a tournament is held among randomly selected individuals (their number is determined by the tournament size) and the best among them is selected into the next generation. The rank-based selection overcomes the problem of fitness proportional selection: premature convergence. Each individual is ranked according to its fitness and consequently selected according to the value of its rank (absolute fitness values are replaced with rank).

DECISION TREES

Decision tree is a typical representative of a symbolic machine learning approach used for the classification of objects into decision classes, where an object is represented in a form of an attribute-value vector ($attribute_1$, $attribute_2$, ... $attribute_N$, *decision class*).[16,17] The attribute values describe the features of an object. The attributes are usually identified and selected by the creators of the data set. The decision class is one special attribute whose value is known for the objects in the learning set and which will be predicted according to the induced DT for all further unseen objects. Normally, the decision class is a feature that could not be measured (e.g., some prediction for the future) or a feature whose measuring is unacceptably expensive, complex, or not known at all.

SIDE BAR 1: Benefits of Using Evolutionary Algorithms with Decision Trees

Evolutionary algorithms produce a solution to optimization problems—in our case, a decision tree (DT) being induced is optimized according to given fitness function. As fitness function can include any measurable entity of a DT, the process of DT induction can be optimized toward different objectives. In this manner, EAs offer great adaptability to a wide range of criteria, limitations, and objectives, including the building of cost-sensitive DTs. EAs are used with DTs primarily to optimize a DT classifier in the learning phase. The power of EAs offers the capability of enhancing the desired characteristics, namely accuracy, sensitivity, and specificity while minimizing the size of the classifier. This eliminates the problem of overfitting, improves results and alleviates end users the burden of understanding EAs to understand the end results.

DTs can be used for two types of problems: classification (the decision class is a discrete variable—a label or category to which the data belongs) and regression (the decision class is a continuous variable).

Examples of attributes-decision class objects are patient's examination results and diagnosis, raster of pixels and the recognized pattern, stock market data and business decision, past and present weather conditions, and the weather forecast.

Formal Definition

Let $A_1, \ldots, A_n, C$ be random variables, where A_i has domain $dom(A_i)$ and C has domain $dom(C)$; we assume without loss of generality that $dom(C) = \{c_1, c_2, \ldots, c_j\}$. A decision tree classifier is a function

$$dt : dom(A_1) \times \cdots \times dom(A_n) \mapsto dom(C)$$

Let $P(A', C')$ be a probability distribution on $dom(A_1) \times \cdots \times dom(A_n) \times dom(C)$ and let $t = \langle t.A_1, \ldots, t.A_n, t.C\rangle$ be a record randomly drawn from P; that is, t has probability $P(A', C')$ that $\langle t.A_1, \ldots, t.A_n\rangle \in A'$ and $t.C \in C'$. We define the misclassification rate R_{dt} of classifier dt to be $P(dt\langle t.A_1, \ldots, t.A_n\rangle \neq t.C)$. The training database D is a random sample from P, the A_i corresponds to the attributes, and C is the decision class.

A decision tree is a directed, acyclic graph T in a form of a tree. Each node in a tree has either zero or more outgoing edges. If a node has no outgoing edges, then it is called a decision node (a leaf node); otherwise, a node is called a test node (or an attribute node). Each decision node N is labeled with one of the possible decision classes $c \in \{c_1, \ldots, c_j\}$. Each

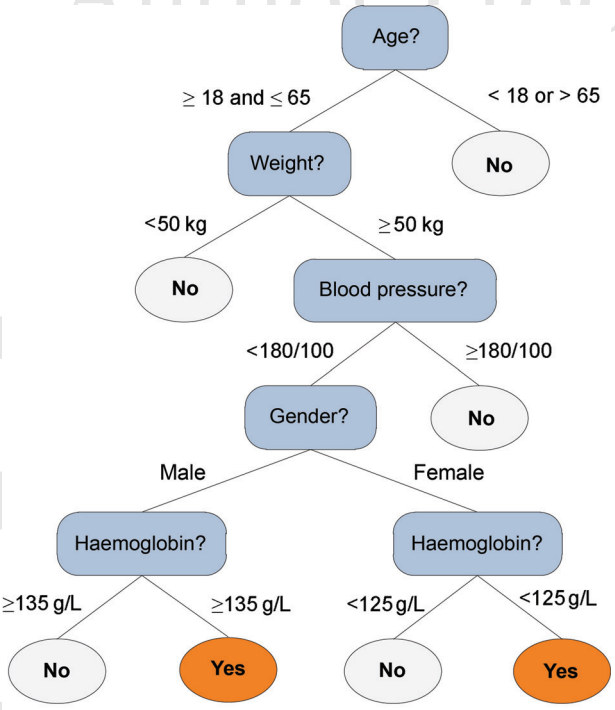

FIGURE 3 | A decision tree that determines whether a person is suited to be a blood .donor

test node is labeled with one attribute $A_i \in \{A_1, \ldots, A_n\}$, i.e. called the splitting attribute. Each splitting attribute A_i has a splitting function f_i associated with it. The splitting function f_i determines the outgoing edge from the test node, based on the attribute value A_i of an object O in question. It is in a form of $A_i \in Y_i$ where $Y_i \subset dom(A_i)$; if the value of the attribute A_i of the object O is within Y_i, then the corresponding outgoing edge from the test node is chosen. An example of a decision tree is shown in Figure 3, in which the decision on whether someone is appropriate for a blood donor is being considered.

The problem of the DT construction is as follows: Given a data set $D = \{t_1, \ldots, t_d\}$, where the t_i are independent random samples from an unknown probability distribution P, find a decision tree classifier T such that the misclassification rate $R_T(P)$ is minimal.

Classical (Statistical) Induction of DT

Inductive inference is the process of moving from concrete examples to general models where the goal is to learn how to classify objects by analyzing a set of instances (already solved cases) whose decision classes are known. Instances are typically represented as attribute-value vectors. Learning input consists of a set of such vectors, each belonging to a known

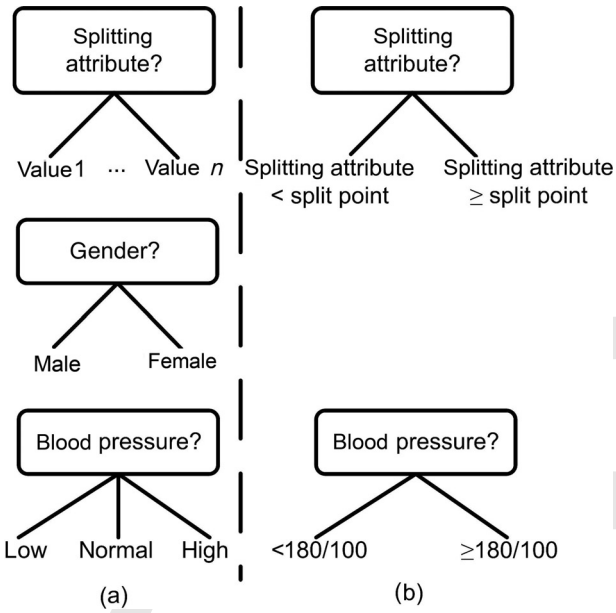

FIGURE 4 | Examples of partitioning a decision tree with regard to data type: (a) discrete value type produces a new branch for each value and (b) continuous type produces two branches based on the chosen split point.

decision class, and the output consists of a mapping from attribute values to decision classes. This mapping should accurately classify both the given instances and other unseen instances.

A decision tree is formalism for expressing such mappings[18] and consists of test nodes linked to two or more subtrees and leafs or decision nodes labeled with a decision class. A test node computes some outcome based on the attribute values of an instance where each possible outcome is associated with one of the subtrees. An instance is classified by starting at the root node of the tree. If this node is a test, the outcome for the instance is determined and the process continues using the appropriate subtree. When a leaf is eventually encountered, its label gives the predicted decision class of the instance. Two basic types of generating split nodes (partitioning) are shown in Figure 4: (a) a discrete-valued tuple and (b) a continuous-valued tuple.

Classically, a DT is built from a set of training objects according to the "divide-and-conquer" principle. When all objects are of the same decision class (the value of the output attribute is the same) then a tree consists of a single node—a leaf with the appropriate decision. Otherwise, an attribute is selected and a set of objects is divided according to the splitting function of the selected attribute. The selected attribute builds an attribute (test) node in a growing DT classifier, and for each outgoing edge from that node the inducing procedure is repeated upon the remaining objects regarding the division until a leaf (a decision class) is encountered.

In 1960s, Hunt et al. introduced CLS (concept learning system)[19] that used heuristic look-ahead to construct trees. CLS was a learning algorithm that learned concepts and used them to classify new cases. CLS was the precursor to DTs, and it led to Quinlan's ID3 system. ID3[20–22] added the idea of using information content to choose the attribute to split. Quinlan later upgraded ID3 with an improved industrial version C4.5 that is still regarded as the reference model to build a DT based on the traditional statistical approach.[18] However, there are many methods for constructing DT classifiers, various splitting criteria, and pruning methodologies.[23]

Variants of DTs

Both algorithms ID3 and C4.5 represent the most common DT approach—the univariate classification trees. They use the statistical calculation of information gain from a single attribute to build a DT. In this manner, an attribute that adds the most information about the decision upon a training set is selected first, and the next one selected is the most informative from the remaining attributes, etc. until a subset of training data is clear enough to be classified with a specific decision class in a leaf node of the DT.

Although the univariate classification trees are used in most applications, there are some extensions and variations of this concept. Unlike a univariate DT, a multivariate DT[24] is not restricted to splits of the instance space that are orthogonal to the features' axes. Utgoff's ID5 was an extension of ID3 that allowed many-valued classifications as well as incremental learning.[25] In 1984, Breiman et al.[26] introduced CART, which, in general, uses the same basic algorithm as Quinlan in ID3 and C4.5. At the decision node level, however, the algorithm becomes extremely complex. CART starts with the best univariate split. It then iteratively searches for perturbations in attribute values (one attribute at a time), which maximize some goodness metric. At the end of the procedure, the best oblique and axis-parallel splits found are compared and the better of them is selected. CART also allowed the induction of a special kind of DTs, namely the regression trees where a tree leaf is a continuous-valued attribute and not a discrete attribute (decision class) as in more traditional classification trees. A particular case of DTs employed to solve regression problems is also model trees,[27] which have the advantage of presenting an interpretable output with an acceptable level of predictive performance.

DTs in Medicine

Decision trees result in comprehensible, small, and efficient classification rules. The results can often be applicable with a printed sheet that is added to a patient chart. To offer widespread use and availability and enable verification of results Cremilleux and Robert[28] offered a general framework for using DTs for medical application. Decision trees have been applied in many areas of medicine: Tsien et al.[29] showed the competitiveness of decision trees for diagnosing myocardial infarction, and Babic et al[30] used fuzzy decision trees in support of breastfeeding, etc. Despite the wide area where DTs have been applied, classical approaches to their induction are not always optimal. Therefore, evolutionary approaches, among others, have been proposed to enhance the decision trees.

EVOLUTIONARY DECISION TREES

DT induction is a complex process. Therefore, deterministic induction approach is not necessarily optimal regarding the quality of obtained DTs and includes several deficiencies. Basically, there are two possibilities to address this problem: to improve a single DT induction process by using alternative approaches or to improve the overall classification performance by using ensemble methods. The main objective of ensemble methods is to annul the deficiencies of a single DT by inducing several DT classifiers and combining them into a single classification model.[31] However, this paper is mainly focused on using evolutionary algorithms to improve the induction of a single DT.

A number of different alternative methods to the induction of DTs have been introduced by various researchers, which try to overcome the deficiencies of classical induction procedures. Most of them are based on soft methods such as evolutionary techniques or neural networks, and sometimes several methods are combined in a hybrid algorithm. Fu et al.[32] proposed evolution of previously developed decision trees (C4.5) to improve performance on large data sets. Evolutionary enhancement of decision trees[33] has been the topic of many research projects.[34,35] A vast number of techniques have also been developed, which help to improve only a part of the DT induction process. Such techniques include evolutionary algorithms for optimizing split functions in attribute nodes, dynamic discretization of attributes, dynamic subset selection of training objects, etc. The three main areas of using evolutionary approaches to enhance decision trees are optimization of training set selection, optimization of the tree construction, and the enhancement of nodes in the tree.

DT Building Support and Induction

Weise and Chiong[36] presented an approach to relax the constraints on the decision expressions (in the nodes of the tree) along with the optimization of the tree shape using genetic programming.[8] The relaxation was aimed at allowing more complex nodes (instead of simple comparisons the decision node can be complex formulae). GP used the following function set: $F = \{+, -, *, \wedge, \vee, \neg, =, >, <, \leq, \geq, \square =$, if-then-else$\}$ and the terminal set was consisted of constants, attribute references, and class identifiers. The optimization process was driven by three objective functions: f_1 is the number of correctly classified samples used in negated form (because of minimization), f_2 is a cost matrix provided a priori (or identity matrix), and f_3 is the size of the classifier (number of nodes in DT). Ensembles of decision trees (evolved according to the fitness functions) are used in a voting process with two variants: (1) the number of votes for each classifier is inversely proportional to the cost of the classifier (in terms of f_2) and (2) the number of votes in inversely proportional to f_1 and f_3. Weise and Chiong compared the results with three well-known approaches: Random-Forrest, DT C4.5, and RepTree (Weka specific). The results showed that the approach is comparable to other approaches often proving to be the most accurate. Q3

The use of evolutionary algorithms to produce the tree induction heuristic (to generate model trees) was also proposed by Barros et al.[37] The work is based on LEGAL-Tree.[38] The approach (E-Motion) addresses the problem of instability when model decision trees are built with greedy algorithms that optimize the standard deviation. Barros et al. tried to provide a middle ground between a model interpretability (model trees built with greedy algorithms) and predictive performance (model trees built by ensemble methods). E-Motion uses GA to evolve a population of individual model trees and aims at finding a global optimum. It was evaluated against M5P (decision tree with linear regression functions at the nodes) and RepTree. It is showed that it to be superior to RepTree in Ref 10 (in regard to root mean-squared error; RMSE) or Ref 9 (in regard to mean absolute error; MAE) UCI[39] data sets. M5P proved better but within statistically insignificant margins.

Andras and Dumitrescu[40] have explored the strength of the evolutionary approach to build decision trees. They use a linear representation of

chromosomes called MEP[41] to represent individual trees in the population.

Poli et al.[42] proposed a combination of genetic programming and simulated annealing for the evolution of DTs. The approach uses a cellular automaton for the representation of individuals. An advantage of cellular automaton use is the consequential parallelization of GP and the avoidance of premature convergence. Experiments have been performed on standard UCI[39] data sets and proved the approach to be on par with C4.5 in regard to the classification error while generating smaller trees that generalize better.

Kretowski and Grzes[43] proposed the evolution of DTs represented in their natural form. The approach called for specialized operators (*CrossTrees* and *MutateNode*). The selection used was linear ranking with elitism. The approach was validated on UCI[39] data sets and proved to produce trees comparable to C4.5 with reduced complexity.

Kretowski and Grzes[44] also proposed a new EA for the induction of mixed decision trees. The proposed system searches for the optimal tree in a global manner: It learns the structure and node tests in one run of the EA. Specialized genetic operators were proposed (to exchange subtrees, prune trees, change node type and tests). Informed mutation reduces unprofitable modifications. The approach was tested on several data sets with promising results.

Czajkowski and Kretowski[45] have proposed a similar approach (to the one mentioned above) for induction of univariate regression trees that associate leaves with simple linear regression models. The intial population is with diverse top–down methods from random subsets of training data. Efficient evolution is supported by specialized operators, Akaike's information criterion is used as the fitness function. The results show that the resulting trees can be significantly less complex with at least comparable performance to classical approaches.

Papagelis and Kalles[46] proposed a GATree methodology of building decision trees using genetic algorithms to directly evolve decision trees. GATree initially builds a set of random minimal binary decision trees (often called decision stumps) that are later combined and modified using mutation–crossover operators. Furthermore, the proposed technique tries to build accurate and simple decision trees; therefore, the fitness function also includes simplicity of a decision tree.

Evaluation of GATree included in comparison to C4.5 and OneR classifiers in the Weka environment.[47] Results using fivefold cross-validation on 13 UCI repository data sets demonstrate superiority of GATree even though the difference with C4.5 was not significant. However, these results were accompanied by extremely small decision trees (C4.5 produced six times bigger trees on average).

The GATree approach was later further enhanced using lossless fitness inheritance.[48] Kalles and Papagelis have presented an extension to a genetic algorithm that evolves binary decision trees to improve the evolution speed by reusing past fitness calculations. Their approach was evaluated on 12 UCI repository data sets, and it shows that savings are substantial and across a diverse number of data sets and experimental configurations with regard to setting of GATree parameters.

GATree has also been used to evolve a population of simple binary DTs using GALIB (a robust C++ library of GA components).[49] The initial population of DTs was generated as simple three node DTs (one attribute node and two leaves). Mutation chooses a random node of a desired tree, and it replaces that node's test value with a new random chosen value. When a random node is a leaf, it replaces the installed class with a new random chosen class. The crossover operator chooses two random nodes and just swaps these nodes' subtrees. The fitness function is balanced between accuracy and size:

$$\text{fitness} = \frac{\text{correctly}^2 + \mathbf{X}}{\text{size}^2 + \mathbf{X}}$$

where $\mathbf{X}$ is an arbitrary big number. When there are many DTs with similar characteristics in a population, their fitness is reduced. The authors tested their approach on 13 UCI data sets and compared the results with C4.5 and OneR on accuracy (average accuracies: C4.5 = 78.46, OneR = 72.64, GATree = 78.75) and tree size (C4.5 = 43.2, GATree = 7.01).

Another approach to evolutionary construction of decision trees, presented by Sprogar et al.,[50] requires minimal human interaction and shows some interesting properties when used on different medical data sets. The algorithm uses an implicit fitness evaluation in the selection phase of a coevolving environment. The proposed algorithm is able to monitor and adjust its own behavior using self-adaptation of evolution parameters. Coevolutionary search is an extension of standard evolutionary search in which the fitness of evolving solutions depends on the state of the other, coevolving individuals rather than a fixed evaluation function. It involves either one or two populations: In the first case, the fitness of an individual is based on competition among individuals in the population, whereas in the second it is based on the individual's behavior in the context of the individuals of another population.[51] The latter type is often

described as "host–parasite" or "predator–prey" coevolution.

The weak point of the proposed algorithm is the fact that it evolves a population of solutions and gives no information to which solution is the best; therefore, it also has no mechanism to select the final solution when the evolution is over. Therefore, the proposed solution still needs manual intervention of the domain expert to select the final decision tree. The algorithm's capability to self-adapt to a given problem is used as a measure to predict whether some data set is just difficult or impossible to analyze using decision trees. The autonomous algorithm on average produces very general solutions or gives no solution if the data set is prone to the overfitting problem.

This study focuses on the medical domain when evaluating the proposed algorithm. It was tested on 10 medical data sets from UCI repository and 2 additional data sets from the medical domain. The selected data sets included from 4 to 101 descriptive attributes, the number of samples ranges from 150 to 48,842, and the majority class occupies 40%–94% of all records. Four data sets define three possible classes, whereas the others represent binary classification problems. Unfortunately, only evaluation using predefined learning and test sets was done instead of more general and accurate cross-validation or hold-out approaches.

Sprogar et al.[52] also presented an interesting approach to genetic induction of vector decision trees. The vector decision tree gives multiple classifications in one single pass (or in one single tree). The main difference to the usual (nonvector) DT is that in the leaf nodes the vector DT suggests several classifications at the same time (a vector of solutions). The authors used a classical GA to induce DTs. The method was applied to a cancer recurrence data set (a nonpublic data set) and compared to C5.0 and discriminant analysis.

Basgalupp et al.[53] recently proposed the so-called lexicographic multiobjective evolutionary induction of decision trees. The proposed LEGAL-Tree avoids the greedy search performed by conventional decision tree induction algorithms and performs instead a global search in the space of candidate decision trees. In addition, the fitness function in LEGAL-Tree is based on lexicographic approach, in which multiple measures are evaluated in order of their priority. The proposed algorithm was evaluated using two most common measures for evaluation of decision trees: classification accuracy and decision tree complexity. Results from seven UCI data sets show that one of the two proposed variants of LEGAL-Tree algorithm obtained significantly better results than the very well-known J48 (implementation of the C4.5 algorithm in the Weka framework) algorithm overall. More precisely, the former obtained statistically significantly simpler decision trees than the latter in three data sets, whereas the opposite was true in just one data set; and such an overall improvement in simplicity was obtained without any significant loss in predictive accuracy in any data set.

Biedrzycki and Arabas[54] also presented a classical method for constructing DTs using a simple GA. First, they defined the space of all possible trees and then GA is used to find good trees by searching that space. The evolutionary process consisted only of mutation—the mutation of a chromosome results in choosing at random a chromosome from its neighborhood, i.e., the mutated tree is either expanded by a nonterminal node or a nonterminal node shrinks from it. There is no crossover. The fitness function considers both accuracy and number of nodes. They compared their EA with J48 and ID3 (accuracy and number of nodes) on six UCI data sets. The best results were obtained with J48 (both in accuracy and number of nodes), followed by EA and then by ID3.

Cha and Tappert[6] have presented a GA for the construction of compact decision trees. They proposed a new scheme to encode and decode a decision trees to and from the chromosome representation. The compactness of the tree is assured by the limitation of the tree height. Usually when applying GAs to DTs directly results in yielding subtrees that are never visited. This problem was addressed with the proposed encoding/decoding scheme. The fitness function of the approach was the sum of depth of the leaf nodes for each instance. Q4

The approaches introduced above are compared in Table 1 with regard to the focus area, algorithm, representation, approach, and the data sets used for evaluation.

Node Optimization

Qiangfu[57] presented an approach that combines decision trees, neural networks (NN), and GA, *neural network tree*. The nodes in the decision tree are neural networks, which are optimized with GA. The decision trees, obtained in this manner are actually modular neural networks with the added restriction of only local decisions. This restriction allows local optimization (of a node) with the use of GA. The primary idea behind the combination of DT and NN is that each neural network is used to extract certain features that are used as decision nodes in the tree. The approach was compared to the standard DT implementation C4.5 and was proven to reduce the number of nodes

TABLE 1 | A Comparison on Focus Area, Algorithm, Representation, Fitness, and Evaluation Data Sets of Approaches to DT Building Support and Induction

Author	Focus Area	Algorithm	Representation	Fitness	Data Sets
Weise and Chiong[36]	Releasing constraints on decision expressions	GP	Function set $F = \{+, -, *, \wedge, \vee, \neg, =, >, <, \leq, \geq, \square =$, if-then-else$\}$ and a terminal set consisting of constants, references to the attributes, and class identifiers[36].	Functions f_1 (correctly classified samples), f_2(a priori cost matrix), and f_3 (a size of classifier) subjected to minimization	URI[39]: Iris, Wine, Heart Disease, Winconsin Breast Cancer, Hepatitis
Barros et al.[37]	Problem of instability when constructing DTs with greedy algorithms	GA: tournament selection, crossover with exchange of subtrees, Mutation: exchange of subtrees or replacement of randomly selected nodes with basic trees	A set of nodes (nonterminal or terminal). Each nonterminal node contains an attribute, each leaf node contains a linear regression model	Two error measures: RMSE and MAE	UCI[39] : AutoMpg, BreastTumor, FishCatch, MachineCPU, Quake, Stock, Strike and Veteran
Andras and Dumitrescu[40]	Evolutionary technique for representing DTs (MEP)	GA	Linear representation of chromosomes called MEP[41]	A number of correctly classified instances	UCI[39]: 11 data sets
Poli et al.[42]	Combination of genetic programming and simulated annealing	GP	Cellular automaton	Classic fitness with "temperature parameter"	UCI[39]: 12 data sets
Kretowski and Grzes[43]	A competitive classifier in terms of accuracy and complexity	EA by framework from in Ref. 55,with specialized operators	Direct representation, actual form	$F = Q_{\text{Rclass}} - L * S$, where Q_{Rclass} is classification quality, S is tree size, and L is importance of complexity(user supplied)	Artificial chessboard data sets and UCI[39] data sets
Kretowski and Grzes[44]	Induction of mixed decision trees	EA	Direct representation	Reclassification quality and DT size	Artificial and real-life data
Czajkowski and Kretowski[45]	Induction of univariate regression trees that associate leaves with simple linear regression models	GA with special operators	Direct representation	Akaike's information criterion	Preliminary experimental validation
Papagelis and Kalles[46]	Overcoming of greedy heuristics and search the DT space in a more natural way	GA with special operators	Direct representation (tree structure)	Balances accuracy and size	UCI[39] : 13 data sets

(*Continued*)

TABLE 1 | Continued

Author	Focus Area	Algorithm	Representation	Fitness	Data Sets
Kalles and Papagelis[48]	Extension of previous work with reusing past fitness calculations	GA with special operators	Direct representation (tree structure)	Balances accuracy and size	UCI[39] : 12 data sets
Papagelis and Kalles[49]	Evolution a population of simple binary DTs	GALIB (C++ library)	Direct representation (tree structure)	Balances accuracy and size	UCI[39] : 13 data sets
Sprogar et al.[50]	Autonomous EA for medical data	Autonomous evolution[56]	Two populations: DT's and samples[51]	Implicit fitness concept	UCI[39] : 10 data sets
Basgalupp et al.[53]	Lexicographic multiobjective evolutionary induction of decision trees	GA with adapted operators	Direct representation allowing multiple-splitting of nodes	Based on lexicographic approach, where multiple measures are evaluated in order of their priority	UCI[39] : 7 data sets
Cha and Tappert[6]	Creation of compact binary decision trees	GA	Encoding scheme	The sum of depth of the leaf nodes for each instance	The approach was not validated on well-known data sets

(not counting the neurons in the nodes) and reduces classification errors.

Cantú-Paz and Kamath[58] present an approach where genetic algorithms are used for selection of splits in oblique DTs. Usual DTs include tests at each node using only one attribute—i.e., equivalent to hyperplanes that are parallel to one of the axes in the attribute space. In contrast to classical DTs, oblique DTs also include hyperplanes that are not axis parallel and therefore include multivariate tests. However, because of difficult interpretability of the results, such tests are not as popular as univariate tests. The greedy search is replaced by two evolutionary algorithms in a study by Cantú-Paz and Kamath. The first one is an extension of OC1 algorithm[59] that uses a $(1 + 1)$ evolution strategy with self-adaptive mutations. It starts with an initial hyperplane using coefficients selected by OC1 and continues with 1000 iterations of coefficients mutations. The second proposed algorithm is based on a simple genetic algorithm with real-valued genes. It uses pairwise tournament selection without replacement, uniform crossover with probability 1.0, and no mutation.

The evaluation of the proposed algorithms was done using seven smaller data sets from UCI repository (two of them from medical domain), two larger UCI data sets, and three artificial data sets with different known characteristics. The results demonstrated the capability of proposed algorithms to find the oblique trees with similar accuracy to OC1, and that it can be done at a competitive cost. The experiments also showed that evolutionary algorithms scale up better than traditional methods to the dimensionality of the data.

The above approaches are compared in Table 2 with regard to the focus area, algorithm, approach they are compared to, fitness functions, and valuation data sets.

Genetic Programming

Genetic programming for hybrid multivariate decision tree consisting of polynomial, fuzzy, and decision tree structures is an approach, as proposed by Mugambi,[60] where oblique decision trees are used instead of classical linear ones. The polynomial nature of these multivariate trees enabled him to perform well in nonlinear territory, whereas the fuzzy members are used to squash continuous variables. By trading-off comprehensibility and performance using a multiobjective genetic programming optimization algorithm, the induced polynomial–fuzzy decision trees are smaller, more compact, and better performer than their linear DT counterparts. Mugambi discusses

TABLE 2 | A Comparison on Focus Area, Algorithm, Approach They Are Compared to, Used Fitness Measure, and Evaluation Data Sets of Approaches to Node Optimization

Author	Focus Area	Algorithm	Compared to	Fitness	Data Sets
Qiangfu[57]	Hybrid DT for node optimization	NNTree (neural network tree)	C4.5	Information gain ration (primary fitness); approximation error (secondary fitness)	UCI[39]: optdigits
Cantú-Paz and Kamath[58]	Selection of splits in oblique DTs with GAs	OC1-ES: extension to OC1 using (1+1) evolution strategy; OC1-GA: extension to OC1 using simple GA; OC1-SA: extension to OC1 using simulated annealing	OC1-AP (OC1 limited to axis-parallel tests); OC1 (with its default parameters); CART-LC	The impurity of a split at each tree node using the twoing rule (the default in OC1)	Smaller data sets from UCI[39]: cancer, diabetes, glass, housing, iris, vehicles, vowel; larger data sets from UCI: optical digit-recognition, pen-based digit-recognition; artificial data sets

Q5

the structural differences between polynomial–fuzzy decision trees and linear decision trees (C4.5) and experimentally compares the size and performance of their models using medical data.

Experiments were conducted on two different medical data sets: diabetes and trauma. Diabetes data set included 2304 records of patients who attended a diabetic clinic. The 29 patient attributes included factors that are associated with diabetes and the type of complications the patients may have suffered in the period they have had diabetes. The class attributes described patients whose diabetic status was likely to deteriorate, leading to complications that are normally associated with the disease. The training set included 1300 cases whereas the test set consisted of the remaining 1004 cases. Trauma data set consisted of 15,000 cases with information on survival of patients admitted to trauma units. Owing to a large number of samples, only 5000 were used for evaluation (3000 for training and 2000 for testing). With only eight attributes, the Trauma data set represents an example of a low-dimensionality data set. Similar to other studies using evolutionary decision trees, it was demonstrated that in terms of classification performance the results of C4.5 are comparable to the results of polynomial–fuzzy decision trees. At the same time, it can be observed that the complexity of built models is lower in the case of polynomial–fuzzy decision trees.

Johansson and Niklasson[61] presented a hybrid method combining genetic programming (GP) and DT learning—DTiGP. The GP implementation of DTiGP is based on an open source framework for evolutionary data mining (G-REX) developed by the authors. The method starts by estimating a benchmark level of reasonable accuracy, based on DT performance on bootstrap samples of the training set. Next, a normal GP evolution is started with the aim of producing an accurate GP—the fitness function considers error rate and size:

$$\text{fitness} = \text{error}_{rate} + \text{error}_{rate} * \text{size} * w.$$

At even intervals, the best GP in the population is evaluated against the accuracy benchmark. If the GP has higher accuracy than the benchmark, the evolution continues normally until the maximum number of generations is reached. If the accuracy is lower than the benchmark, two things happen: (1) the fitness function is modified to allow larger GPs and able to represent more complex models, and (2) a DT with increased size and trained on a bootstrap of the training data is injected into the population.

The experiments show that the hybrid solution of injecting DTs into a GP population gives synergetic effects, producing results that are better than using either technique separately. The authors compared their method DTiGP with jaDTi (an open source implementation based on C4.5), J48 (from Weka), and normal GP (which uses the same settings and fitness function as DTiGP). The results, from 18 UCI data sets, show that the proposed method clearly outperforms normal GP and is significantly better than the

TABLE 3 | A Comparison of Genetic Programming Approached with Regard to Focus Area, Algorithm, Approach They Are Compared to, Used Fitness Measure and Evaluation Data Sets

Author	Focus Area	Algorithm	Compared to	Fitness	Data Sets
Mugambi[60]	Hybrid multivariate oblique DT induction approach for the induction of polynomial-fuzzy DTs	Multiobjective GP optimization algorithm	C4.5	Multiobjective optimization: tree size and classification performance (ROC, sensitivity and specificity)	UCI: diabetes, trauma
Johansson and Niklasson[61]	Hybrid method for learning DTs using GP	DTiGP—a hybrid algorithm for injecting DTs into a GP population	jaDTi (open-source implementation of C4.5), J48 (from Weka) and normal GP (using the same settings and fitness function as DTiGP)	Weighted sum of error rate and size	18 UCI data sets: breast- cancer, breast-w, colic, credit-a, credit-g, cylinder bands, diabetes, ecoli, glass, haberman, heart-Cleveland, heart-statlog, hepatitis, iris, labor, liver-disorders, lymph, tic-tac-toe,

Q6

standard DT algorithm (jaDTi = 75.8, GP = 77.3, DTiGP = 79.5, J48 = 76.8).

The above approaches are compared in Table 3 with regard to the focus area, algorithm, approach they are compared to, used fitness measure, and evaluation data sets.

Application of EDTs in Medicine

The main criteria for any model developed for medicine is the transparency and a clear understanding of the inner workings of the model. Many approaches where genetic algorithms are used for optimization of performance achieve highly accurate results, although many of them are highly complex. The complexity results in the use of the model as a black box. This problem was addressed by Motsinger[62] et al., who proposed the use of grammatical evolution to optimize decision trees (GEDT). The approach focused on the analysis of genetic association studies to detect gene–gene interactions.

The presence of microcalcifications (MCs), clusters of tiny calcium deposits that appear as small bright spots in a mammogram, has been considered as a very important indicator for breast cancer diagnosis. In Ref 63, the authors use evolutionary algorithms to adjust the fitness value for each bright spot in the process of MC detection with the combined use of decision trees and genetic algorithms. The method used is to award the associate chromosomes that have detected MCs successfully and give a penalty to the chromosomes that are associated with false MCs. The authors demonstrated that the incorporation of the knowledge-discovery mechanism into the genetic algorithm is able to eliminate the false MCs and produce improved performance compared with the conventional GA methods. Furthermore, the experimental results show that the proposed method provides a promising and generic approach for the development of computer-aided diagnosis for breast cancer.

In Ref 64, the authors analyzed the data set of extracted hemodynamic features through computational intelligence techniques for identification of potential prognostic factors for periventricular leukomalacia (PVL) occurring in neonates with congenital heart disease. The extracted features were used as inputs to two classifiers, namely multilayer perceptron and probabilistic neural network in combination with genetic algorithms for selection of the most suitable features predicting the occurrence of PVL. The selected features were next used as inputs to a decision tree algorithm for generating easily interpretable rules of PVL prediction. The approach combines the generalization capability of the computational intelligence (CI) based feature selection approach and generation of easily interpretable classification rules of DT. The combination of CI techniques with DT gave

substantially better test prediction performance than using CI and DT separately.

Microarray data provides information on gene expression levels of thousands of genes in a cell in a single experiment. A DNA microarray is a powerful tool in the diagnosis of cancer. Numerous efforts have been made to use gene expression profiles to improve precision of a tumor classification. In Ref 65, a comparison between class prediction accuracy of two different classifiers, genetic programming and genetically evolved decision trees, was carried out using the best 10 and best 20 genes ranked by the *t*-statistic and mutual information. Genetic programming proved to be a better classifier for this data set based on area under the receiver operating characteristic curve (AUC) and total accuracy using mutual information based feature selection.

Q7

In Ref 66, the authors present a hybrid intelligence method that integrates genetic algorithm and decision tree learning techniques for knowledge mining of an *in vitro* fertilization (IVF) medical database. The proposed method can not only assist the IVF physician in predicting the IVF outcome but also find useful knowledge that can help the IVF physician to tailor the IVF treatment to the individual patient with the aim of improving the pregnancy success rate.

In Ref 67, authors presented a method for automatic rules induction using evolutionary induction of decision trees and automatic programming. The proposed algorithm was applied to a cardiovascular data set consisting of different groups of attributes that should possibly reveal the presence of some specific cardiovascular problems in young patients. Beside good classification results, a possible new medical knowledge in the field of pediatric cardiology was automatically discovered.

Turney[3] proposed a cost-sensitive version of decision trees where genetic algorithm is used to optimize the tree and improve the average cost of classification error and cost of tests (obtaining variable measurements). The proposed ICET (Inexpensive Classification with Expensive Test) approach was tested on five UCI data sets. The paper demonstrates that accuracy is not the only performance measure that should be used for classification in medical domain.

Chai et al.[68] proposed Binary Tree–Genetic Algorithm (BTGA), a linear decision tree structure to construct piecewise linear classifiers. A GA was used at each nonterminal node for searching the optimal (maximum impurity reduction) linear decision function. BTGA demonstrated a much lower cross-validation misclassification rate. A modified BTGA was applied to pap smear cell classification that offers the sensitivity needed for automated prescreening of pap smear slides.

Zorman et al.[69] have performed a comparison of different strategies of decision tree induction on a medical data set containing 2673 cases of orthopedic fracture data. The cases were described by 23 attributes (6 continuous and 17 discrete), and the decision had three possible values (successful, failed, unverified failure). The decision trees were built with six different approaches; four classical, a hybrid of neural networks and decision trees, and an evolutionary approach. The evolutionary approach represented individuals in the population as decision trees. This approach has the benefit of making all intermediate solutions directly usable, and the direct representation eliminates the loss of information that could occur with an indirect representation. The approach however requires some additional work with the definition of genetic operators.

The comparison (Table 4) of approaches was done on the measurements of prediction accuracy, sensitivity to failure, specificity, size of the tree, and overall accuracy. The evolutionary approach showed the best performance while keeping the size of the tree small and easily understandable. The authors showed that building decision trees with the evolutionary approach provides the capability of personalizing, or providing priorities on which performance metric is the most important for the research being done. Contrary to this, weighting the cases in statistical induction methods improves sensitivity, reducing specificity, and not improving the overall accuracy.

The above approaches are compared to the focus area, algorithm, representation, fitness, and evaluation data sets.

Q8

SIDE BAR 2: Advantages of EDTs over Other Approaches

Real-world problems that require classification are diverse and complex, especially in medicine and nursing. These areas favor the decision tree because of the prediction power and the ease and simplicity of understanding how the decisions were performed in each step of the classification process. The evolutionary approach to decision tree induction improves the induction by prioritizing performance metrics, personalization of DT construction, and eliminating classical constraints such as numeric attributes and missing values.

TABLE 4 | A Comparison on Focus Area, Algorithm, Representation, Fitness, and Evaluation Data Sets of Application to EDT's in Medicine

Author	Focus Area	Algorithm	Representation	Fitness	Data Sets
Motsinger et al.[62]	Interpretability of gene–gene interaction classification	Grammatical Evolution Decision Trees	The grammar can be represented by the tuple of nonterminals, terminals, production rules (terminals and nonterminals mapped into rules) and root (nonterminal) node.	Arithmetic average of sensitivity and specificity.	100 simulated gene–gene interaction data sets containing 250 samples and 100 attributes each.
Peng et al. [63]	Combination of decision trees and genetic algorithms for computer-aided breast cancer diagnosis	The genetic algorithm is used to search for the bright spots in mammogram and a knowledge-discovery mechanism (DT) is integrated to improve the performance of the GA.	Feature space extracted from mammogram images.	Grouping of bright spots from mammogram images.	Data set based on images of microcalcifications (MCs) in mammogram for breast cancer prognosis.
Samanta et al.[64]	Using genetic algorithm to select features that are later used as an input to DT predicting PVL.	Multilayer perceptron (MLP) and probabilistic neural network (PNN) in combination with genetic algorithms (GA) for selection of classifier parameters.	The number of neurons in the hidden layer for MLP and the RBF width (σ) for PNN.	Overall accuracy for different numbers of selected features.	PVL occurrence in neonates with congenital heart disease data set.
Podgorelec et al.[67]	Evolutionary induction of decision trees applied to pediatric cardiology.	Evolutionary induction of decision trees and automatic extraction of classification rules	Nodes of the DT.	A weighed sum of classification performance (classification accuracies of each decision class), tree size, and number of different attributes used.	Pediatric cardiology data set focusing on specific cardiovascular problems.
Turney[3]	Optimization of cost-sensitive DTs using genetic algorithms applied to different UCI data sets.	Evolution of a population of biases for a decision tree induction algorithm.	Attribute weights for attribute selection in decision tree node induction.	Average cost of classification when using the decision tree, including both the costs of tests (features, measurement) and the costs of classification errors.	UCI[39]: BUPA Liver, Heart Disease, Hepatitis Prognosis, Pima Indians Diabetes and Thyroid Disease.

(*Continued*)

TABLE 4 | Continued

Author	Focus Area	Algorithm	Representation	Fitness	Data Sets
Chai et al.[68]	Genetic algorithm based linear decision tree applied to pap smear cell classification problem.	GA-based linear classifier search function is used in each nonterminal node.	Coefficients of the linear classifier for each node.	Impurity estimation at each nonterminal node.	Pap smear cell classification data.
Zorman et al.[69]	Comparison of different DT induction algorithms including evolutionary approach with application on orthopedic fracture data.	Multipopulation evolutionary evolution is used for decision tree optimization.	DT representation-decision trees may be seen as a kind of simple computer programs (with attribute nodes being conditional clauses and decision nodes being assignments)	Accuracy, sensitivity and specificity of the classification of training objects.	Orthopedic fracture data containing 2673 cases and 23 attributes.

CONCLUSIONS

A great amount of research has been directed toward evolutionary development of decision trees. The primary motive is to enhance the performance of classic decision trees. Decision trees offer many benefits over other classifiers: They do not require extensive knowledge in the target domain (they are appropriate for exploratory knowledge discovery), they are appropriate for high-dimensional data, and they offer an intuitive representation that is easy to understand and comprehend. Classifiers are evaluated on the basis of accuracy, speed, robustness, scalability, and interpretability. The standard method of evaluation is based on public data sets, like UCI[39] databases. Evolutionary approaches combined with decision trees are aimed at maintaining their good characteristics (easy to understand, no extensive knowledge on domain required, high-dimensional data, etc.). The evolutionary optimization is aimed at training set selection and optimization, node optimization, and tree induction optimization.

Optimization of individual nodes of the tree has shown that evolution of individual nodes results in highly complex nodes. Consequently, a review of individual decisions (in each node) is far more difficult. A straightforward condition is easier to understand than a complex neural network or a complicated equation. Therefore, it seems that evolutionary optimization of nodes reduces the primary features of a DT classifier (readability) and transforms it to a black box classifier. Black box classifiers are inherently less appropriate for medical applications.

Two major research directions of using evolution to enhance DT performance for medical applications are optimization of training set selection and tree construction. Evolutionary optimization (EO) of training sets has been researched in terms of large data sets scaling problems, selection of subsets of training data, evolving representations of training sets, imbalanced data sets, and training data feature sampling. Most of this research have shown that evolutionary optimization of training set improves the performance of the resulting classifier. Moreover, evolutionary optimization of the training set does not influence the important parts of a decision tree classifier (comprehensiveness, high dimensionality, etc.).

EO of the induction of the tree has been researched in terms of: relaxation of decision expressions, optimization of tree shape, decision tree instability (because of greedy algorithms that optimize standard deviation), construction of the entire tree with evolutionary approaches (linear representation of chromosomes represents individual trees in an evolving population of trees), representation with cellular automation with simulated annealing for the evolution of individuals, evolution of DTs in their natural form (using special genetic operators), using self-adaptation of evolutionary parameters to perform coevolutionary search, lexicographic

multiobjective evolutionary induction of decision trees, etc. Results are of course heavily dependent on each research approach. However, many experiments have shown that the resulting trees are smaller with at least comparable performance when compared with classically built decision trees.

REFERENCES

1. Graham VJ. Missing data analysis: making it work in the real world. *Ann Rev Psychol* 2009, 60:549–576.
2. Donders RAT, van der Heijden GJMG, Stinjen T, Moons KGM. A gentle introduction to imputation of missing values. *J Clin Epidemiol* 2006, 59:1087–1091.
3. Turney P. Cost-sensitive classification: empirical evaluation of a hybrid genetic decision tree induction algorithm. *J Artif Intell Res* 1995, 2:369–409.
4. Kretowski M, Grzes M. Evolutionary induction of decision trees for misclassification cost minimization. *Lecture Notes Comput Sci* 2007, 4431:1–10.
5. Freitas A. Building cost-sensitive decision trees for medical applications. *AI Commun* 2011, 24:285–287.
6. Freitas A, Costa-Pereira A, Brazdil P. Learning cost-sensitive decision trees to support medical diagnosis. In: Nguyen T., ed., *Complex Data Warehousing and Knowledge Discovery for Advanced Retrieval Development: Innovative Methods and Applications*. 2010, 287–307.
7. Holland JH. *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press; 1975.
8. Koza JR. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press; 1992.
9. Fogel LJ, Owens AJ, Walsh MJ. *Artificial Intelligence through Simulated Evolution*. New York: John Wiley & Sons; 1966.
10. Schwefel HP. *Evolution and Optimum Seeking*. New York: John Wiley & Sons; 1995.
11. Barros RC, Basgalupp MP, de Carvalho ACPLF, Freitas AA. A survey of evolutionary algorithms for decision-tree induction. systems, *IEEE Trans Man Cybernet, Part C* Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp = & arnumber = 5928432&isnumber = 4359283 (Accessed 2011).
12. Goldberg DE. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley; 1989.
13. Rawlins GJE. ed. *Foundations of Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann; 1991.
14. Whitley D, Rana S, Heckendorn R. Representation issues in neighborhood search and evolutionary algorithms. In: *Genetic Algorithms and Evolution Strategy in Engineering and Computer Science*. West Sussex, UK: John Wiley & Sons Ltd.; Chapt 3, 39–58.
15. Mitchell M. *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT Press; 1996.
16. Podgorelec V, Zorman M. Decision trees. *In: Encyclopedia of Complexity and Systems Science*. New York: Springer; 2009, 2:1826–1845.
17. Rokach L, Maimon OZ. Data mining with decision trees: theory and applications. In: *Series in Machine Perception and Artificial Intelligence*. 2008; 69.
18. Quinlan JR. *C4.5: Programs for Machine Learning*. San Francisco: Morgan Kaufmann; 1993.
19. Hunt EB, Marin J, Stone PT. *Experiments in Induction*. New York: Academic Press; 1966, 45–69.
20. Quinlan JR. Discovering rules by induction from large collections of examples. In: *Expert Systems in the Micro Electronic Age*. 1979, 168–201.
21. Quinlan JR. Induction of decision trees. *Mach Learn* 1986, 1:81–106.
22. Quinlan JR. Simplifying decision trees. *Int J Man-Mach Stud* 1987, 27:221–234.
23. Rokach L, Maimon O. Top-down induction of decision trees Classifiers—a survey. *IEEE Trans Syst Man Cybernet Part C* 2005, 35:476–487.
24. Brodley CE, Uttgof PE. Multivariate decision trees. *Mach Learn* 1995, 19:45–77.
25. Utgoff PE. Incremental induction of decision trees. *Mach Learn* 1989, 4:161–186.
26. Breiman L, Friedman JH, Olshen RA, Stone CJ. *Classification and Regression Trees*. Wadsworth; 1984.
27. Quinlan JR. Learning with continuous classes. In: Proceedings of the 5th Australian Joint Conference on AI, 1987. 221–234.
28. Crémilleux B, Robert C. A theoretical framework for decision trees in uncertain domains: application to medical data sets, artificial intelligence in medicine. *Lecture Notes Comput Sci* 1997, 1211:143–156.
29. Tsien CL, Fraser HS, Long WJ, Kennedy RL. Using classification tree and logistic regression methods to diagnose myocardial infarction. *Medinfo* 1998(pt 1):493–497.
30. Babic SH, Kokol P, Stiglic MM. Fuzzy decision trees in the support of breastfeeding. In: Proceedings of the 13th IEEE Symposium on Computer-Based Medical Systems, 2000. 7–11.
31. Breiman L. Random forests. *Mach Learn* 2001, 45:5–32.

32. Fu Z, Golden BL, Lele S, Raghavan S, Wasil EA. A genetic algorithm-based approach for building accurate decision trees. *Inform J Comput* 2003, 15:3–22.
33. Wu S. *Better Decision Tree from Intelligent Instance Selection: A New Instance Selection Method Based on Genetic Algorithm for Optimizing Decision Trees*. Saarbrücken, Germany: VDM Verlag; 2009.
34. Koza J. Concept formation and decision tree induction using the genetic programming paradigm. In: Parallel Problem Solving from Nature. *Lecture Notes Comput Sci* 1991, 496:124–128.
35. Nikolaev N, Slavov V. Inductive Genetic Programming with Decision Trees. In: Machine Learning: ECML-97. *Lecture Notes Comput Sci* 1997, 1224:183–190.
36. Weise T, Chiong R. (2010). Evolutionary data mining approaches for rule-based and tree-based classifiers. In: 9th IEEE International Conference on Cognitive Informatics (ICCI), 2010. 696–704.
37. Barros RC, Basgalupp MP, Ruiz DD, de Carvalho ACPLF, Freitas AA. Evolutionary model tree induction. In: *Proceedings of the 2010 ACM Symposium on Applied Computing—SAC '10*. New York: ACM Press; doi: 10.1145/1774088.1774327.
38. Basgalupp MP, Barros RC, de Carvalho ACPLF, Freitas AA, Ruiz DD. LEGAL-tree: a lexicographic multi-objective genetic algorithm for decision tree induction. ACM Sympos Appl Comput (SAC). 1085–1090.
39. Frank A, Asuncion A. UCI machine learning repository. Irvine, CA: University of California, School of Information and Computer Science. Available at: http://archive.ics.uci.edu/ml .
40. Andras J, Dumitrescu D. Evolving orthogonal decision trees. *Informatica* 2003, XLVIII:33–44.
41. Oltean M. Multiexpression programming. Technical report. Romania: Babes-Bolyai University. Available at http://www.mep.cs.ubbcluj.ro/MEP.pdf.
42. Poli R, Banzhaf W, Langdon W, Miller J, Nordin P, Fogarty T, et al. Genetic Programming and simulated annealing: a hybrid method to evolve decision trees—genetic programming. *Lecture Notes Comput Sci* 1802:294–303.
43. Kretowski M, Grześ M. Global learning of decision trees by an evolutionary algorithm. *Inform Process Security Syst* 2005:401–410.
44. Kretowski M, Grzes M. Evolutionary induction of mixed decision trees. *Int J Data Warehousing Mining* 2007, 3:68–82.
45. Czajkowski M, Krętowski M. Globally induced model trees: an evolutionary approach. In: Parallel Problem Solving from Nature–PPSN XI. *Lecture Notes Comput Sci* 2011, 6238:324–333.
46. Papagelis A, Kalles D. GA tree: genetically evolved decision trees. In: 12th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'00), 2000. 203–206.
47. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH, The WEKA data mining software: an update. *SIGKDD Explorations* 11:2009.
48. Kalles D, Papagelis A. Lossless fitness inheritance in genetic algorithms for decision trees, *Soft Comput* 2010, 14:973–993.
49. Papagelis A, Kalles D. Breeding decision trees using evolutionary techniques. In: Proceedings of the 18th International Conference on Machine Learning (ICML-2001), 2001. 393–400.
50. Sprogar M, Sprogar M, Colnaric M. Autonomous evolutionary algorithm in medical data analysis. *Comp Methods Program Biomed* 2005, 80(suppl 1):29–38.
51. Pagie L, Mitchell M. A comparison of evolutionary and coevolutionary search. In Belew RK, Juill H, eds. *Coevolution: Turning Adaptive Algorithms upon Themselves*. San Francisco, CA; 2001, 20–25.
52. Sprogar M, Lenic M, Alayon S. Evolution in medical decision making. *J Med Syst* 2002, 26:479–489.
53. Basgalupp M, Barros R, Carvalho A, Freitas A, Ruiz D. Lexicographic multiobjective evolutionary induction of decision trees. *Int J Bio-Inspired Comput* 2009, 1:105–117.
54. Biedrzycki R, Arabas J. Evolutionary and greedy exploration of the space of decision trees. In: Arabas J, ed. Evolutionary Computation and Global Optimization. Prace Naukowe, Elektronika Warsaw, Poland: Warsaw University of Technology Publishing House; 2006, 479–489.
55. Šprogar M. Autonomous evolutionary methods and classification models. PhD thesis. Maribor, Slovenia: Faculty of Electrical Engineering and Computer Science; 2002.
56. Michalewicz Z. *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin, Germany: Springer; 1996.
57. Qiangfu Z. Evolutionary design of neural network tree-integration of decision tree, neural network and GA. In: *Proceedings of the 2001 Congress on Evolutionary Computation* IEEE. pp. 240–244. doi: 10.1109/CEC.2001.934395.
58. Cantú-Paz E, Kamath C. Using evolutionary algorithms to induce oblique decision trees. In: Whitley D, Goldberg DE, Cantú-Paz E, Spector L, Parmee L, Beyer H-G, eds. *Proceedings of the Genetic and Evolutionary Computation Conference 2000*. San Francisco, CA: Morgan Kaufmann; 2000, 1053–1060.
59. Murthy SK, Kasif S, Salzberg S. A system for induction of oblique decision trees. *JAIR* 1994, 2:1–32.
60. Mugambi E. Polynomial-fuzzy decision tree structures for classifying medical data. *Knowledge-Based Syst* 2004, 17:81–87.
61. Johansson U, Niklasson L. Improving GP classification performance by injection of decision trees. *Comput Intell* 2010, 18–23.

62. Motsinger-Reif AA, Deodhar S, Winham SJ, Hardison NE. Grammatical evolution decision trees for detecting gene-gene interactions. *BioData Mining* 2010, 3:8.
63. Peng Y, Yao B, Jiang J. Knowledge-discovery incorporated evolutionary search for microcalcification detection in breast cancer diagnosis. *Artif Intell Med* 2006, 37:43–53.
64. Samanta B, Bird GL, Kuijpers M, Zimmerman RA, Jarvik GP, Wernovsky G, Clancy R-R, Licht DJ, Gaynor JW, Nataraj C. Prediction of periventricular leukomalacia. Part II: Selection of hemodynamic features using computational intelligence. *Artif Intell Med* 2009, 46:217–231.
65. Kulkarni A, Kumar N, Ravi V, Murthy US. Colon cancer prediction with genetics profiles using evolutionary techniques. *Expert Syst Appl* 2011, 38:2752–2757.
66. Guh RS, Jackson Wu TC, Weng SP. Integrating genetic algorithm and decision tree learning for assistance in predicting in vitro fertilization outcomes. *Expert Syst Appl* 2011, 38:4437–4449.
67. Podgorelec V, Kokol P, Stiglic M, Hericko M, Rozman I. Knowledge discovery with classification rules in a cardiovascular dataset. *Comput Methods Program Biomed* 2005, 80(suppl 1), S39–S49.
68. Chai B, Huang T, Zhuang X, Zhao Y, Sklansky J. Piecewise linear classifiers using binary tree structure and genetic algorithm. *Pattern Recogn* 1996, 29:1905–1917.
69. Zorman M, Podgorelec V, Kokol P, Peterson MGE, Sprogar M, Ojstersek M. Finding the right decision tree's induction strategy for a hard real world problem. *Int J Med Inform* 2001, 63:109–121.

Queries

Q1: AU: Is EDT abbreviated correctly for evolutionary decision trees

Q2: AU: References are renumbered to appear in a sequence. Please confirm.

Q3: AU: Please provide symbol in place of □ here and in Table 1.

Q4: AU: At Ref 6., Cha and Tappert is not listed. If required, please provide complete details of it.

Q5: AU: The sense of "twoing" is not clear. Please check.

Q6: AU: Please expand ROC.

Q7: AU: Please expand AUC.

Q8: AU: Please verify that the edited sentence conveys the intended sense.

Q9: AU: Please provide name and location of the publisher.

Q10: AU: Please provide the correct URL so that this reference is accessible to the readers. Also provide month and date in access date.

Q11: AU: Please provide name and location of the publisher also if name of the editors required.

Q12: AU: Please provide name and location of the publisher also if name of the editors required.

Q13: AU: Please provide name and location of the publisher.

Q14: AU: Please provide page range and year of publication, if possible.

Q15: AU: Please provide page no. and vol. no.

Q16: AU: Please provide access date.

Q17: AU: Please provide technical report no. and access date.

Q18: AU: Please provide year of publication and check the page range for correctness.

Q19: AU: Please verify that this reference is OK as typeset.

Q20: AU: Please provide year and location of the publisher.

Q21: AU: Please provide abbreviated journal title.

Q22: AU: Please provide vol. no.