Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Operations Research 33 (2006) 3185–3202
computers & operations research
www.elsevier.com/locate/cor

Diversification for better classification trees

Zhiwei Fu[a], Bruce L. Golden[b,*], Shreevardhan Lele[b], S. Raghavan[b], Edward Wasil[c]

[a]*Fannie Mae, 4000 Wisconsin Avenue NW, Washington DC 20016, USA*
[b]*R.H. Smith School of Business, University of Maryland, College Park, Maryland, MD 20742, USA*
[c]*Kogod School of Business, American University, Washington DC 20016, USA*

Available online 22 April 2005

Abstract

Classification trees are widely used in the data mining community. Typically, trees are constructed to try and maximize their mean classification accuracy. In this paper, we propose an alternative to using the mean accuracy as the performance measure of a tree. We investigate the use of various percentiles (representing the risk aversion of a decision maker) of the distribution of classification accuracy in place of the mean. We develop a genetic algorithm (GA) to build decision trees based on this new criterion. We develop this GA further by explicitly creating diversity in the population by simultaneously considering two fitness criteria within the GA. We show that our bicriterion GA performs quite well, scales up to handle large data sets, and requires a small sample of the original data to build a good decision tree.

Keywords: Data mining; Classification trees; Genetic algorithm

1. Introduction

Decision trees are a popular technique for classification. The main reason behind their popularity appears to be their relative advantage in terms of interpretability. Their popularity has also been aided by easily available implementations such as CART (see [1]) and C4.5 (see [2]). Hastie et al. [3] provides a comprehensive survey of tree methods. The construction of decision trees usually proceeds in two steps: (a) growing the tree with the objective of maximizing classification accuracy on a training set, and (b) pruning the tree to avoid the problem of overfitting. Li et al. [4] discuss various current issues in pruning.

* Corresponding author. Tel.: +1 301 405 2232; fax: +1 301 405 3364.
E-mail address: bgolden@rhsmith.umd.edu (B.L. Golden).

0305-0548/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2005.02.035

In Fu et al. [5], we showed that it could be advantageous to use a genetic algorithm (GA) for the purpose of constructing an accurate decision tree. In particular, it was shown that instead of using the entire training set data to construct a single tree using an algorithm like C4.5, it is better to partition the available data into subsets, construct a C4.5 tree on each subset, and to then use this set of trees as the initial population in a GA. This technique presented in Fu et al. [5] is a genetic algorithm for intelligent trees (GAIT).

However, as is well known in the statistical literature, the accuracy of a classifier is a random variable, and is best described via a distribution, rather than a single estimated parameter such as the mean. In Fu et al. [6], GAIT was improved upon in two significant respects. First, the objective of the GA, called its fitness function, was changed from a simple average accuracy criterion to a probabilistic criterion that used the complete distribution of the classification accuracy. By fine-tuning this criterion, it is possible to construct trees that are specifically tailored to the decision maker's level of risk aversion. For example, a risk-averse decision maker may wish to obtain a tree whose accuracy has a low probability of falling below a relatively low threshold (a risk averse objective), while a risk-seeking decision maker may wish to obtain a tree whose accuracy has a high probability of falling above a relatively high threshold (a risk seeking objective). Second, the GA was modified to allow for simultaneous dual objectives, e.g., a risk-averse objective and a risk-seeking objective. This modification produced a diversity of trees at each stage. It was shown that such diversity led to improved trees, even if the eventual objective is somewhere between the risk-averse objective and the risk-seeking objective.

In this paper, we extend our prior work in a couple of significant ways. First, we model the decision maker's attitude to classification accuracy using a bilinear loss function. For such a loss function, the optimal criterion is an appropriate *percentile* of the accuracy distribution. In particular, the median accuracy criterion is motivated along with the two extreme values, the minimum and the maximum. Second, we show that even if the decision maker is solely interested in maximizing the median accuracy, it pays to run a GA with diverse objectives such as maximizing the minimum accuracy (a risk-averse objective) and maximizing the maximum accuracy (a risk-seeking objective). The diversity of the trees so obtained is shown to be advantageous in eventually creating a tree with a high median accuracy.

The rest of this paper is organized as follows. In Section 2, we review GAIT, the basic algorithm that was developed in Fu et al. [5]. Section 3 discusses risk preferences that may arise in the construction of a classifier, and the accompanying fitness criteria. The need to estimate percentiles of the accuracy distribution is motivated here. In Section 4, we provide numerical results in constructing decision trees tailored to different risk preferences using five different data sets (including three from the publicly available UCI repository, see Blake and Merz [7]). In Section 5, we consider simultaneous dual criteria and provide numerical results that show that it indeed pays to consider a wide diversity of trees. Scaling issues are presented in Section 6, while Section 7 offers concluding remarks.

2. A GA for intelligent trees

In this section, we review GAIT, the GA for classification trees developed by Fu et al. [5]. First, we briefly discuss related work in the use of evolutionary algorithms for constructing classification trees.

GAs and genetic programming have been used in a wide variety of data mining applications like clustering and rule discovery (see [8,9] for nice surveys). There have been several applications of genetic programming to constructing decision trees (see [10–13]), but only a few papers addressing the use of GAs

for designing decision trees. These include some early work using small data sets by Ryan and Rayward-Smith [14] and Kennedy et al. [15], our earlier work described in Fu et al. [5,6], and some recent work by Cantú-Paz and Kamath [16] on constructing oblique decision trees. (In oblique decision trees, tests at nodes contain multiple attributes or are multivariate tests, while in axis-parallel decision trees, which our algorithm deals with, tests at nodes of the decision tree deal with a single attribute or is a univariate test.) Genetic programming methods use variable length programs that can represent complex structures while GAs use fixed-length strings for representation. This allows genetic programming methods to perform more complex operations than GAs, but they result in more computationally intensive search techniques. Consequently, when compared to GAs, genetic programming methods require more CPU time, more memory, and, in addition, scalability is a concern.

GAIT begins with an initial population of classification trees. These trees constitute the first generation of the GA. GAIT then applies the genetic operations of crossover and mutation to generate new trees. It evaluates the fitness of all trees in a generation to determine the set of trees that survive to the next generation (those that do not survive are deleted). GAIT repeats this procedure for a finite number of generations, or until some specified level of convergence is obtained. We now provide further details on the various aspects of GAIT.

Initial population. We split the data set available for tree generation purposes into two parts. A *training set* and a *scoring set*. From the training set we generate the initial set of trees as follows. Suppose the training set consists of N_t points. Then GAIT generates the initial set of k trees by randomly sampling N_t/k points, either with or without replacement from the training set. (N_t is selected to be an integer multiple of k.) A tree is generated using any classification algorithm on this sample of N_t/k points. This procedure is repeated until k trees are generated. In our work, we use the popular C4.5 algorithm (see [2]) to generate the set of initial trees, and sample with replacement from the training set.

Crossover. Fig. 1(a) shows the crossover operation. From two parent trees, our crossover operation randomly selects a node in each parent, and exchanges the subtrees rooted at these nodes to obtain two children trees. A standard roulette wheel parent selection mechanism is used to ensure that the most fit current trees are given more reproductive chances than other current trees. Crossover is accepted with a probability equal to the crossover rate.

Mutation. Following crossover, a mutation operator is applied. Fig. 1(b) shows the mutation operation. The mutation operator consists of the exchange of subtrees within a tree. Two nodes are randomly selected within the tree and the subtrees rooted at each of these nodes are exchanged. Obviously, to make sure the operation works, we ensure that neither of the nodes selected is in the other node's subtree. Mutation is accepted with a probability equal to the mutation rate.

Fitness function. A GA with an elitist strategy scores the population of feasible solutions in a generation using a fitness criterion to determine the set of solutions that survive to the next generation. In our application of GAIT [5], we use the classification accuracy of a tree on the scoring set as the fitness criterion. However, GAIT can easily be adapted to use any other fitness criterion. In the next section, we examine alternate criteria for evaluating the performance of a classification tree, and use these as the fitness criterion within GAIT.

Logic checks. As presented, our crossover and mutation operations may result in trees with logically inconsistent rules. This does not affect the classification accuracy of the tree since logically inconsistent rules never classify any data points (since no data points would satisfy the logically inconsistent conditions of the rule). However, in terms of the quality of the trees generated by the GA, it is beneficial to prune the tree by deleting branches corresponding to logically inconsistent rules. As a result, within our GA,

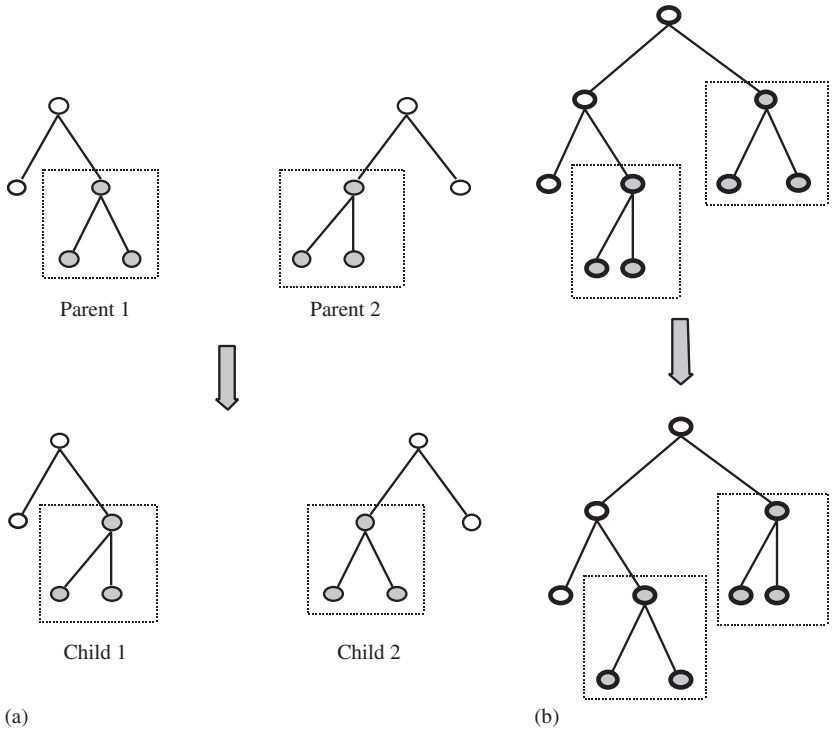

Fig. 1. (a) Crossover operation. (b) Mutation operation.

we check and correct these logical inconsistencies within the trees (see Fig. 2(a)). Observe that the logic checks could be performed either immediately after the crossover or mutation operations, or delayed until the end of a generation. In terms of the trees generated, the two approaches are equivalent. However, it is computationally more efficient to delay the logic checks to the end of a generation.

Pruning. At the end of GAIT, we also employ standard pruning criteria (using the scoring set) to increase the ability of the tree generated by GAIT to classify data sets other than the data set on which it is generated (see Fig. 2(b)). Pruning starts at the bottom of the tree and works its way up to the root of the tree. Each internal node (i.e., nonleaf node) is replaced with a leaf if its replacement results in a decrease in classification accuracy that is smaller than a pre-specified value (e.g., 0.5%).

We note that for simplicity we have restricted our figures to binary decision trees. However, GAIT may be applied to any poly-ary decision tree (for a more elaborate discussion on this issue see [5]). In Fu et al. [5], our computational implementation was restricted to binary decision trees. With further developments in our code since then, our implementation of GAIT can handle poly-ary trees.

In Fu et al. [5], we demonstrated that GAIT generates high-quality decision trees and works effectively on very large data sets. We also showed that its sampling-based technique outperforms the common-sense strategy of constructing a single decision tree using all the available data. Boosting and bagging are ensemble-based methods, developed over the last decade, for combining a set of classifiers to arrive at a more accurate classifier (see [17] for a nice survey). Boosting algorithms (see [18,19]) create an initial set of trees by adaptively changing the distribution of the training set. These trees are then combined according

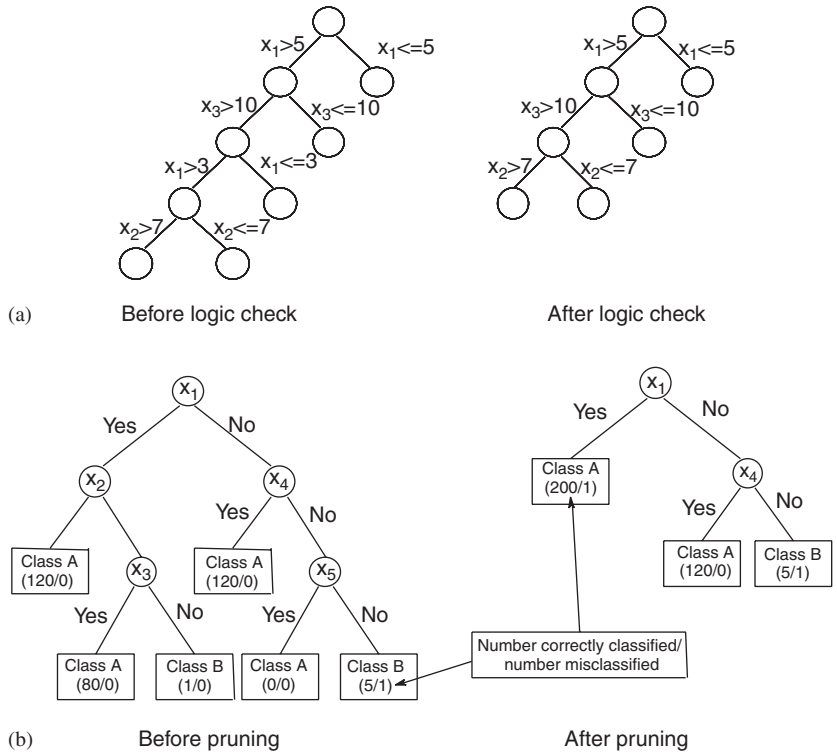

Fig. 2. (a) Logic check. (b) Pruning.

to a system of weights. With bagging (see [20]), the initial set of trees is produced from bootstrapped samples obtained from a fixed training set. These are combined using a majority voting rule. GAIT resembles bagging in the creation of the initial set of trees. But, instead of creating bootstrapped samples from the training set, it partitions the training set into sub-samples. Additionally, GAIT's strategy of combining and evolving the initial set of trees is very different from the majority voting rule of bagging. One advantage of GAIT over ensemble methods such as boosting and bagging is that the end product of GAIT is a single decision tree. In boosting and bagging, the entire ensemble of initial trees is to be retained for future classification. Further, in Fu et al. [5], we showed GAIT improves the accuracy that is obtained from a majority voting rule such as bagging.

Of course, in addition to accuracy, stability and simplicity of decision trees are also of acute interest to decision makers. These measures typically decrease with an increase in tree size (depth) and complexity (number of leaf nodes). If two decision trees employ the same kind of branch tests and have the same prediction accuracy, the one with fewer leaf nodes is typically preferred (see [21]).

In our work (see [22]), we compared the stability and simplicity of the final decision trees generated by our algorithms against the stability and simplicity of the decision trees generated by other algorithms. Our algorithms are designed to generate decision trees from significantly smaller data sets than the original data sets, since we use a sampling and partitioning approach. In other words, our decision trees are not generated directly from the original (very large) data sets. Consequently, the depth and number of leaf nodes in our decision trees tend to be smaller than those generated by competing algorithms.

In addition, since GAIT employs logical checking and pruning to explicitly control tree growth, one would expect the resulting trees to be lean and stable and this has been confirmed in a variety of computational experiments (see [22]).

3. Fitness criteria

In this section, we motivate some new criteria for evaluating the performance of a tree and show that selecting a specific criterion depends on the risk preference of the evaluator (decision maker). Consider a tree that is to be used to classify a set of n items that have been randomly selected from a given population. Suppose the misclassification costs of the various classes are equal, so that the classification accuracy is simply the overall proportion of correctly classified items. We wish to use the classification accuracy of the tree on a random sample of n items as a fitness measure of the tree. Such a measure is a random variable; it has a probability distribution. Typically, the mean value of this distribution is estimated.

We propose an alternative to using the mean accuracy as the performance measure of a tree. In place of the mean, we investigate the use of various percentiles of the distribution of the classification accuracy. In particular, we focus on the *minimum*, the *maximum*, and the *median* of the accuracy distribution.

If the accuracy distribution is symmetric, then its mean and median coincide, and any estimator for the mean is also an estimator for the median. If the accuracy distribution is not symmetric, then, its mean and median are not equal and the median can be motivated as a reasonable measure of central tendency in its own right. The median is a particular case of a percentile (the 50th). From the viewpoint of statistical decision theory, regardless of the symmetry of the accuracy distribution, it is possible to motivate any percentile as the optimal parameter to be estimated by choosing an appropriate bilinear loss function for the estimation process (e.g., see [23]). This is in contrast to the square-error loss function, for which the optimal decision is to estimate the mean.

3.1. The Bilinear loss function

Let θ denote the classification accuracy of the tree on a randomly selected set of n items. θ has a probability distribution. Let T be the estimated value of the accuracy that should be attached to the tree. If the loss function that is faced in estimating θ by T is of the square-error form $L(T, \theta) = (T - \theta)^2$, then, the optimal value of T that minimizes the expected loss, $E[L(T, \theta)]$, is simply the mean of the accuracy distribution, $E[\theta]$. However, if the loss function is of the bilinear form

$$L(T, \theta) = \begin{cases} (T - \theta) & \text{if } T \geqslant \theta, \\ c(\theta - T) & \text{if } T < \theta, \end{cases}$$

where $c \geqslant 0$ is the asymmetry parameter, then, the optimal value of T that minimizes $E[L(T, \theta)]$ is the $100[c/(1 + c)]$th percentile of the distribution of θ.

Clearly, if the bilinear loss function is symmetric (i.e., if $c = 1$), then, the optimal decision is to estimate the median accuracy (i.e., the 50th percentile). If $c = 0$, the optimal decision is to estimate the minimum accuracy (i.e., the 0th percentile), while if $c \to \infty$, the optimal decision is to estimate the maximum accuracy (i.e., the 100th percentile).

3.2. Selecting the asymmetry parameter

The selection of the specific form of the bilinear loss function should be guided by the larger context in which the estimated accuracy percentile is to be used. In this paper, we use the estimated accuracy percentile in order to rank and select the best tree(s) among k competing trees. Such a step is taken in creating each generation of the GA described in Section 2. The appropriate accuracy percentile is estimated for each tree and the tree with the maximum estimated value of this accuracy percentile is deemed the best.

Suppose we use the bilinear loss function with $c = 0$. Then, we must select the tree with the maximum estimated value of the minimum accuracy as the best tree. In doing so, we are employing a risk-averse strategy for selecting among trees. This is because we are focusing on the least desirable outcome, and selecting among the competing trees that tree whose least desirable outcome is best. Thus, using the bilinear loss function with a small value of c (close to 0) is equivalent to using a risk-averse criterion for estimating the fitness measures of the competing trees.

On the other hand, suppose we use the bilinear loss function with $c \to \infty$. Then, we must select the tree with the maximum estimated value of the maximum accuracy as the best tree. Here, we are employing a risk-seeking strategy for selecting among trees because we are focusing on the most desirable outcome, and selecting among the competing trees that tree whose most desirable outcome is best. Thus, using the bilinear loss function with large values of c is equivalent to using a risk-seeking criterion for estimating the fitness measures of the competing trees.

As a third option, we can use the symmetric bilinear loss function, i.e., where $c = 1$. Then, we must select the tree with the maximum estimated value of the median accuracy as the best tree. By analogy with the previous two strategies, such a selection strategy could be deemed as a risk-neutral strategy, and using the symmetric bilinear loss function is thus equivalent to using a risk-neutral criterion for estimation.

Thus, the choice of the asymmetry parameter, c, should be guided by the risk preference of the decision maker, with respect to the estimation process. In Section 4, we present algorithms tailored for different types of risk preferences. In Section 5, we demonstrate that even if the actual objective is to create a tree with a high median accuracy (i.e., if the decision maker is risk-neutral), it still pays to consider trees that score high on risk-averse and risk-seeking criteria.

3.3. Estimating the percentiles

After a particular percentile of the accuracy distribution is selected by the decision maker as optimal with respect to his/her risk preference, the percentile must be estimated from the available data. Methods for estimating the percentiles include (i) using the sample percentiles of the training set that was used to construct the classifier, (ii) using the sample percentiles from a hold-out set (sometimes called a scoring set) that is drawn from the same population as the training set, but is independent of the training set, (iii) resampling methods such as the different variants of cross-validation and bootstrapping.

We use the scoring set approach to obtain the accuracy distribution. A scoring set containing m data sets, each containing n points, is drawn randomly from the total data available for creating the classifiers, and is set aside from the remaining data that are used for training. The accuracy of each tree created as part of GAIT is noted on each of the m scoring sets. These m sample accuracies provide an estimate of the accuracy distribution. The percentiles of this empirical distribution provide estimates of the percentiles

of the accuracy distribution. Clearly, m must be chosen to be sufficiently large in order to obtain accurate estimates of the percentiles.

As noted earlier, if the accuracy distribution is perfectly symmetric, then, the mean accuracy is the same as the median accuracy. This common parameter can then be estimated by either the sample mean or the sample median. However, for heavy-tailed symmetric distributions, the sample median should be preferred as it is a more stable estimator than the sample mean (see [24] for details).

4. GA: different risk preferences

In this section, we report on the computational experiments using GAIT and the new fitness scoring function. Our implementation of GAIT is coded in Microsoft Visual C++ 5.0 and run on a PC with a 500 MHz Pentium II processor and 128 MB of RAM.

4.1. Data sets

We first describe each of the five data sets used in our computational experiments.

Transportation. This data set is from a leading firm in the transportation services industry. The firm has gathered information on whether or not a customer reuses its services after a marketing promotion and would like to identify patterns among the repeat customers. The transportation marketing data set has demographic and usage information on 11 key variables for approximately 440,000 customers (78.37% are not repeat customers and 21.63% are repeat customers).

Adult. This data set is from the UCI machine learning repository (see [7]). This data set consists of observations on 48,842 adult individuals along 14 demographic variables. The objective is to predict one of two income categories for an individual based on the given demographic variables. The two income categories are low (for individuals earning less than $50,000) and high (for individuals earning greater than $50,000). The proportion of the majority class (low income) in the data set is 75.22%.

Financial. This data set was obtained from a direct marketing campaign at a major financial services firm. This data set consists of over two million customer records. The objective is to predict whether or not a customer will accept a credit card promotion that is being offered within a certain window of time. There are 66 predictor variables describing each customer's demographics, credit history, and prior behavior with respect to promotions. Of these 66 variables, 62 are categorical and 4 are numerical. The proportion of the majority class (those who do not accept the promotion) is 83.50%.

Mushroom. This data set from the UCI machine learning repository includes descriptions of hypothetical samples corresponding to 23 species of mushrooms. Each species is identified as edible or poisonous. The *National Audubon Society Field Guide to North American Mushrooms* [25], from which this data set is constructed, clearly states that there is no simple rule for determining the edibility of a mushroom. There are a total of 8124 instances with 22 attributes. The proportion of the majority class (edible) in the data set is 51.8%.

Letter. This data set is from the UCI machine learning repository. The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into

a range of integer values from 0 through 15. Observe that, compared to the four earlier data sets, the classification task is non-binary (i.e., the letter is to be identified as one of 26 classes).

4.2. Experimental design

For each of the transportation, adult, financial, and letter data sets, we randomly select a training set of 10,000 points ($N_t = 10,000$), a scoring set of 4000 points ($N_s = 4000$), and a test set of 4000 points. The test set is not used to train or score decision trees. From the 8124 points in the mushroom data set, we randomly select a training set of 4000 points, a scoring set of 2000 points, and a test set of 2000 points. Again, the test set is not used to train or score decision trees.

We randomly sample with replacement from the training set to generate 200 unique decision trees. Recall from Section 2 that each tree will be generated using C4.5 on a sample of N_t/k points (i.e., 50 points for the transportation, adult, financial, and letter data sets, and 20 points for the mushroom data set). For the transportation, adult, financial, and letter data sets, we partition the scoring set into 40 subsets of 100 points each and calculate the accuracy of each of the 200 decision trees on each subset (i.e., $m = 40, n = 100, k = 200$). We sort the accuracies of the 200 decision trees and select the 50 trees with the best *percentile* accuracies to form the initial population for our GA. Specifically, if we are interested in maximizing the 0th percentile accuracy of the classification tree (i.e., maximizing the minimum classification accuracy of the classification tree) we select the 50 trees based on their minimum classification accuracy on the scoring set. At each step in GAIT, the fitness score of a tree is computed by calculating the *appropriate percentile* classification accuracy of the tree on the scoring set. In the case of the mushroom data set, we partition the scoring set into 20 subsets of 100 points (i.e., n is the same as the other data sets).

We run our GA for 50 generations with a crossover rate of 1.00 and a mutation rate of 0.01 and save the 50 best trees at each generation. We perform 10 replications that result in 10 best final trees. For all data sets, except mushroom, we partition the test set into 40 subsets of 100 points and calculate the accuracy of each of the 10 trees on the 40 subsets. For the mushroom data set, we partition the test set into 20 subsets of 100 points and calculate the accuracy of each of the 10 trees on the 20 subsets.

Since the fitness function depends on the percentile value selected, for convenience, we denote the GA with the new classification function by GAK, where K refers to the percentile value. For example, GA0 is the GA that uses the minimum (0 percentile) of the accuracy distribution. However, for the rest of this paper, we will use the more easily interpretable GAMin when $K = 0$, GAMedian when $K = 50$, and GAMax when $K = 100$.

4.3. Computational results

In Table 1, we report the computational results for the GA with the three different percentile values: 0, 50, and 100 (i.e., GAMin, GAMedian, and GAMax) on the five data sets. The results for each GA are averages on the subsets from the test set over 10 replications. For example, we ran the GAMin algorithm on the transportation data set, generated a best final tree, pruned this best final tree on the scoring set, then evaluated the tree on each of the 40 test subsets, and calculated average results. We repeated this 10 times and averaged over all 10 replications. (In other words, in order to summarize performance with respect to minimum accuracy over all 10 replications, we simply report the average.) The average

Table 1
Computational results for the GA with the (a) Min criterion, (b) Median criterion, and (c) Max criterion

(a) Data set		GAMin	C4.5	*t*-stat	*p*-value
Transportation	Min accuracy (0th percentile)	0.6998	0.6900	6.2963	0.0001
	Running time (minutes)	14.23	1.08		
Adult	Min accuracy (0th percentile)	0.7031	0.6920	2.8503	0.0095
	Running time (minutes)	18.05	1.83		
Financial	Min accuracy (0th percentile)	0.7625	0.7540	2.7362	0.0115
	Running time (minutes)	21.32	3.83		
Letter	Min accuracy (0th percentile)	0.7695	0.7380	43.8233	0.0000
	Running time (minutes)	16.65	1.63		
Mushroom	Min accuracy (0th percentile)	0.9194	0.9010	26.8075	0.0000
	Running time (minutes)	8.02	0.67		
(b) Data set		GAMedian	C4.5	*t-stat*	*p*-value
Transportation	Median accuracy (50th percentile)	0.7770	0.7670	5.8824	0.0001
	Running time (minutes)	13.24	1.08		
Adult	Median accuracy (50th percentile)	0.8276	0.8050	9.8230	0.0000
	Running time (minutes)	18.12	1.83		
Financial	Median accuracy (50th percentile)	0.8857	0.8720	7.0476	0.0000
	Running time (minutes)	21.12	3.83		
Letter	Median accuracy (50th percentile)	0.7912	0.7610	22.3927	0.0000
	Running time (minutes)	16.13	1.63		
Mushroom	Median accuracy (50th percentile)	0.9440	0.9160	16.4738	0.0000
	Running time (minutes)	8.30	0.67		
(c) Data set		GAMax	C4.5	*t-stat*	*p*-value
Transportation	Max accuracy (100th percentile)	0.8345	0.7920	22.4868	0.0000
	Running time (minutes)	14.19	1.08		
Adult	Max accuracy (100th percentile)	0.8451	0.8380	3.0525	0.0069
	Running time (minutes)	17.92	1.83		
Financial	Max accuracy (100th percentile)	0.9048	0.8910	6.9819	0.0000
	Running time (minutes)	22.10	3.83		
Letter	Max accuracy (100thpercentile)	0.8037	0.7800	19.6052	0.0000
	Running time (minutes)	15.78	1.63		
Mushroom	Max accuracy (100th percentile)	0.9500	0.9380	9.5534	0.0000
	Running time (minutes)	8.58	0.67		

Average results over 10 replications are reported on the tests sets for the best final tree that the algorithm generates. The results for C4.5 are from one run on the training and scoring set. The paired difference *t*-test is for the improvement of GAMin, GAMedian, and GAMax, respectively, over C4.5.

minimum classification accuracy over all 10 runs for GAMin on the transportation data set is 0.6998 (see Table 1(a)).

For comparison purposes, we ran C4.5 on each of the five data sets after combining the training set and the scoring set. Thus, in the case of all data sets except mushroom, C4.5 generates a single tree on a

14,000-point data set, for which we obtained the minimum, median, and maximum classification accuracies on the 40 subsets of the test set. For the mushroom data set, C4.5 generates a single tree on a 6000-point data set and the minimum, median, and maximum classification accuracies are obtained on the 20 subsets of the test set. We conducted a paired-difference t-test for the improvement of GAMin, GAMedian, and GAMax, respectively over C4.5 on the test sets when C4.5 is run on the entire training and scoring set. The results of these tests are also given in Table 1. We see that GAMin generates trees with significantly better minimum classification accuracy than C4.5 (p-value 0.0115 or smaller), GAMedian generates trees with significantly better median classification accuracy than C4.5 (p-value 0.0001 or smaller), and GAMax generates trees with significantly better maximum classification accuracy than C4.5 (p-value 0.0069 or smaller). On average, for all data sets except mushroom which is smaller, one run of GAMin, GAMedian, and GAMax takes 17 and a half minutes, while C4.5 takes about two minutes to generate a tree from each of the 14,000-point data sets.

In the next section, we describe a refinement to our GA procedure that yields significant improvements in the percentile classification accuracy of the trees.

5. Diversification: it pays to consider alternate criteria

In this section, we investigate whether introducing diversity, by fitness function perturbation, results in better classification trees. Many researchers (see [26–28]) point out the benefits of introducing population diversity within a GA. However, it is often difficult to quantify the benefits of introducing diversity into genetic algorithms. In Fu et al. [6], we demonstrated the benefits of introducing diversity while considering a composite fitness function consisting of the mean and standard deviation of the classification accuracy. We continue with this line of investigation, in the context of the percentile classification accuracy. Specifically, we consider the effect of simultaneously relying on two percentile criteria (Min and Max) that are, for the moment different from our final percentile criterion (Median).

5.1. Bicriterion GA

We now describe an adaptation of our GA to handle two fitness criteria. The bicriterion genetic algorithm (BGA) makes a simple change to the basic GAIT algorithm to allow the algorithm to be simultaneously mindful of two objectives while determining the trees that survive and are carried over to the next generation.

Specifically at each generation, BGA maintains two lists—an X list consisting of trees ranked by percentile criterion X (for example the Min criterion) and a Y list consisting of trees ranked by percentile criterion Y (for example the Max criterion). Suppose the number of trees that survives a generation is N. Then the length of each list is also N. This ensures that the number of distinct trees in these two lists is at least N and no more than $2N$. If the trees on the two lists are distinct, then we select the N trees that survive by choosing the top $N/2$ trees on the X list and the top $N/2$ trees on the Y list.

In the situation where the trees on the two lists are not distinct, we use a selection algorithm identical to the stable marriage algorithm (see [29]). Let RX and RY denote the ranked lists of trees that survive a generation. At the beginning of each generation both the lists are empty and the maximum size of both lists is $N/2$. We then select the first item on the X list and add it, in ranked order, to the RX list if the tree is not on the RX list, or on the RY list, and the RX list is not full. Next, we select the first item on the Y list

Table 2
Computational results for the bicriterion GA using Min and Max as the two criteria. (a) Average results are reported on the test sets for the best final tree that the algorithm generated for the Min, Max, and Median criteria. (b) Computational results for paired difference *t*-test for the improvement of BGA(Min,Max) over GAMin, GAMax, and GAMedian, respectively

(a) BGA(Min,Max)

Data set	Criterion	Min	Max	Median	Time (min)
Transportation	Percentile score	0.7097	0.8635	0.7800	70.29
Adult	Percentile score	0.7145	0.8896	0.8647	24.02
Financial	Percentile score	0.7765	0.9105	0.8974	81.69
Letter	Percentile score	0.7788	0.8108	0.8036	19.84
Mushroom	Percentile score	0.9303	0.9616	0.9530	11.49

(b) BGA(Min,Max)

Comparison to criterion data set	GAMin		GAMax		GAMedian	
Criterion	Min		Max		Median	
Data set	*t*-stat	*p*-value	*t-stat*	*p*-value	*t*-stat	*p*-value
Transportation	2.5075	0.0167	6.2398	0.0001	2.2500	0.0255
Adult	3.7851	0.0022	15.8827	0.0000	9.4170	0.0000
Financial	2.8278	0.0099	4.3281	0.0010	6.7789	0.0000
Letter	6.9213	0.0000	3.8335	0.0020	7.5866	0.0000
Mushroom	6.9380	0.0000	6.8123	0.0000	4.9992	0.0004

and add it, in ranked order, to the *RY* list if the tree is not on the *RY* list, or on the *RX* list, and the *RY* list is not full. The procedure continues by considering the second item on the *X* and *Y* lists, the third item on the *X* and *Y* lists, and so on, until both the *RX* and *RY* lists are full.

In the final generation of BGA, we output the best tree as scored by the *X* percentile criterion, and the best tree as scored by the *Y* percentile criterion. In our experiments we set *X* to the Min criterion, and *Y* to the Max criterion. Additionally, since we wish to investigate the benefits of diversity, we also output the best tree (from the trees in the final generation of BGA) as scored by the Median criterion.

We compared BGA using the Min and Max criteria to GAMin, GAMax, and GAMedian by conducting an identical set of experiments to those described in Section 4. These results are summarized in Table 2. To make clear Min and Max are the two criteria being used within BGA, we will denote the bicriterion GA with respect to the Min and Max criteria, by BGA(Min,Max).

For all five data sets in Table 2 results are averages on the test sets over 10 replications. We ran BGA(Min,Max) and, at the end of the algorithm, from the final generation, obtained three trees. The best tree with respect to the Min criterion, the best tree with respect to the Max criterion, and the best tree with respect to the Median criterion. Each of these trees was then evaluated on the test set to determine average summary measures for each tree. We performed these calculations for each of the 10 replications and averaged over them to obtain the summary measures reported in Table 2(a).

Comparing the results in Table 2(a) with those in Table 1 reveals some interesting observations. It seems that BGA(Min,Max) generates trees that score better on the percentile score for *all three* scoring criteria (Min, Max, and Median). For example, the best Max trees generated by BGA(Min,Max) on the transportation data set have a greater classification accuracy than GAMax (0.8635 vs. 0.8345). Since the

Table 3
Computational results for BGAC. Paired difference *t*-test for the improvement over BGA(Min,Max)

BGAC				
Data set	Median accuracy	Time (min)	*t*-stat	*p*-value
Transportation	0.7830	70.13	2.2500	0.0255
Adult	0.8710	23.87	2.8425	0.0097
Financial	0.9010	82.30	2.9459	0.0082
Letter	0.8160	23.67	5.9489	0.0001
Mushroom	0.9590	13.12	3.1779	0.0056

algorithm does not consider the Median criterion until the final generation, this shows that diversity has allowed the discovery of better trees than GAMin, GAMax, and GAMedian. Notice, that diversity has also helped the Min and Max criteria. We have obtained better Min trees (than GAMin) by introducing diversity by keeping the good Max trees in the population. But these improvements have come at an expense, as the running time of BGA is significantly greater than GAMin, GAMax, and GAMedian.

To test the validity of our observations we ran a paired difference *t*-test on the results of BGA(Min,Max) over GAMin, GAMax, and GAMedian, respectively. These are reported in Table 2(b). We see that the percentile score improvements obtained by BGA(Min,Max) over GAMin, GAMax, and GAMedian are significant (*p*-values are 0.0255 or smaller).

5.2. *Bicriterion GA with convergence*

We now describe a modification to BGA that results in a significant improvement in the quality of the trees. The idea is to guide BGA towards the Median criterion, to hopefully find even better classification trees (with respect to the Median criterion).

We have three percentile values. A starting X value, a starting Y value, and a final F value. Each generation of BGAC is identical to BGA, except that we modify the X and Y values at each generation. We increase X and decrease Y so that they converge to F at the final generation of BGAC. The increase in percentile value for X in each generation is given by $(F - X)$/(number of generations). The decrease in percentile value for Y in each generation is given by $(Y - F)$/(number of generations). Notice that in the final generation, both X and Y are equal to F, and the best trees on both the X and Y lists are identical and are the best trees with respect to the F criterion.

We ran an identical set of experiments on BGAC, as we did on BGA. We started with $X = 0$, i.e., the Min criterion, $Y = 100$, i.e., the Max criterion, and set $F = 50$, i.e., the Median criterion. The results of these experiments for all five data sets are shown in Table 3. We see that BGAC generates trees that have greater median accuracy than BGA. The running time of BGAC is comparable to BGA. To test the validity of these conclusions, we ran a paired difference *t*-test on the results of BGAC and BGA. These are also reported in Table 3. We see that BGAC generates trees with significantly greater median accuracy (*p*-values are 0.0255 or smaller).

So far, in our experiments, the computational times of BGAC and BGA are comparable, while GA is faster than BGA, and C4.5 is faster than GA. Before drawing any final conclusions about the computational times, we note that it is important to investigate the performance on larger data sets. That is, to see how

the algorithms scale up. The next section reports on computational experiments to understand how the different GAs scale to handle larger training and scoring sets.

6. Experiments with scaling

In this section, we consider the three large data sets—transportation, adult, and financial—and increase the size of the training set and the size of the scoring set for C4.5 and the three algorithms GAMedian, BGA, and BGAC in order to investigate issues of scalability. As before, the median accuracy measures are obtained from the test set which is not used in the training and scoring of the algorithms.

6.1. Experimental design

In the scaling experiments for the transportation data set of approximately 440,000 data points, we randomly select 4000 points that are set aside to form our test set. We then generate a series of training and scoring combinations to assess scalability as shown in the size columns of Table 4. For each combination, the training set and scoring set are selected at random from the remaining 436,000 points.

Next, we partition each scoring set into a number of subsets. Each subset will include exactly 100 points. For example, for 3% of the original data set with a scoring set of 4000 points, there are 40 scoring subsets of 100 points each. For 25% of the original data set with a scoring set of 32,000 points, there are 320 scoring subsets of 100 points each. For 51% of the original data set with a scoring set of 64,000 points, there are 640 scoring subsets of 100 points each, and so on. In the same way, we partition the test set of 4000 points into 40 subsets of 100 points each.

As we scale up the training set from 10,000 points to 310,000 points, each tree of the initial 200 trees (produced by C4.5) is generated using a larger number of data points. For instance, when the training set contains 10,000 points, each initial tree is generated using 50 points. When the training set contains 80,000 points, each initial tree is generated using 400 data points, and so on for the remaining three training sets.

For each scaling experiment on each training and scoring combination, we record the median accuracy on the test set and computing time for C4.5, GAMedian, BGA, and BGAC. With respect to C4.5, we use the training set and scoring set data as input and compute the median accuracy on the test set. Although the sizes of the training, scoring, and test sets for the experiment with 3% of the data are the same as in Sections 4 and 5, we perform the experiment afresh in this section. Each experiment is replicated 10 times and the performance measures are computed on the test set of 4000 points.

The scaling experiments on the adult and financial data sets are conducted in a similar manner to the transportation data set.

6.2. Experiment results and analysis

In Tables 4–6, we present the median accuracy results for C4.5, GAMedian, BGA, and BGAC and computation times on the transportation, adult, and financial data sets, respectively. We observe that median accuracy increases as the size of the training and scoring combination increases. Additionally, BGAC outperforms BGA which outperforms GAMedian which outperforms C4.5 with respect to the median accuracy. To better illustrate these differences, in Fig. 3, for the transportation data set, we

Table 4
Computational results for five training and scoring combinations based on the average of 10 replications for C4.5, GAMedian, BGA, and BGAC on the transportation data set

Size			C4.5		GAMedian		BGA		BGAC	
%	Training	Scoring	Median accuracy	Time (min)	Median accuracy	Time (min)	Median accuracy	Time (min)	Median Accuracy	Time (min)
3	10,000	4000	0.7670	1.12	0.7772	13.28	0.7800	72.04	0.7830	71.40
25	80,000	32,000	0.7732	35.30	0.7848	21.36	0.7913	80.20	0.7957	80.96
51	160,000	64,000	0.7783	90.52	0.7868	34.16	0.7934	93.51	0.7976	94.10
77	240,000	96,000	0.7814	168.44	0.7878	51.27	0.7940	113.63	0.7981	114.31
99	310,000	124,000	0.7825	416.46	0.7880	70.05	0.7942	138.20	0.7986	139.74

Table 5
Computational results for five training and scoring combinations based on the average of 10 replications for C4.5, GAMedian, BGA, and BGAC on the adult data set

Size			C4.5		GAMedian		BGA		BGAC	
%	Training	Scoring	Median accuracy	Time (min)	Median Accuracy	Time (min)	Median accuracy	Time (min)	Median accuracy	Time (min)
32	10,000	4000	0.8060	1.85	0.8280	18.82	0.8640	24.56	0.8712	25.90
48	15,000	6000	0.8140	17.72	0.8586	23.16	0.8778	29.90	0.8847	31.24
64	20,000	8000	0.8220	35.63	0.8688	28.15	0.8810	37.64	0.8866	35.46
80	25,000	10,000	0.8250	65.92	0.8705	39.71	0.8818	48.32	0.8872	46.77
96	30,000	12,000	0.8260	107.78	0.8710	49.64	0.8822	59.98	0.8880	53.38

Table 6
Computational results for five training and scoring combinations based on the average of 10 replications for C4.5, GAMedian, BGA, and BGAC on the financial data set

Size			C4.5		GAMedian		BGA		BGAC	
%	Training	Scoring	Median accuracy	Time (min)	Median accuracy	Time (min)	Median accuracy	Time (min)	Median accuracy	Time (min)
1	10,000	4000	0.8720	3.84	0.8856	20.90	0.8974	80.55	0.9040	82.32
6	80,000	32,000	0.8890	40.10	0.9024	30.44	0.9157	89.39	0.9278	91.24
11	160,000	64,000	0.8990	104.98	0.9121	44.21	0.9208	103.22	0.9324	108.14
17	240,000	96,000	0.9045	190.23	0.9148	59.32	0.9224	128.32	0.9344	124.45
22	310,000	124,000	0.9064	440.12	0.9157	81.03	0.9232	151.10	0.9351	150.20

plot the median classification accuracy of each algorithm against the size of the training and scoring combination as a percentage of the size of the data. We see that GAMedian, BGA, and BGAC clearly outperform C4.5. We also observe that median classification accuracy jumps most dramatically when training and scoring size increases from 3% to 25%. Beyond that point, median accuracy levels off. It

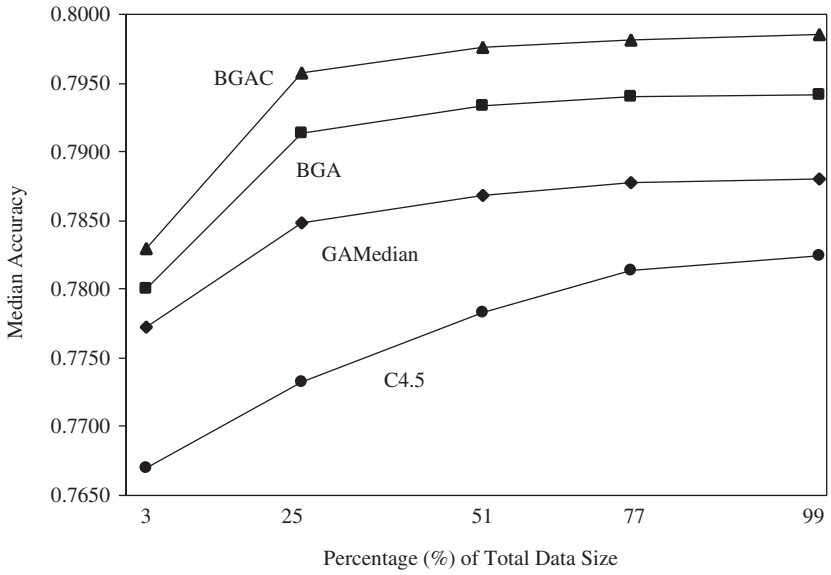

Fig. 3. Median classification accuracy of C4.5, GAMedian, BGA, and BGAC on five training and scoring combinations for the transportation data set.

is important to remark that, using only 25% of the data, GAMedian, BGA, and BGAC outperform C4.5 using 99% of the data. As noted earlier, BGAC outperforms BGA which outperforms GAMedian.

In comparing the average computation times for the experiments, we see that the computation time increases when the size of the training and scoring combination increases for C4.5, GAMedian, BGA, and BGAC. BGAC and BGA require more computation time than GAMedian, while BGA and BGAC take nearly the same amount of computation time. C4.5 has, by far, the fastest growth rate in computing time. Fig. 4 illustrates these differences for the transportation data set.

Two observations are evident from these experiments. First, GAMedian, BGA, and BGAC generate much more accurate trees than C4.5, and their computation times increase linearly as the size of the training and scoring combination increases. Second, GAMedian, BGA, and especially BGAC require only a small percent of the data in order to generate high-quality decision trees.

7. Conclusions

In this paper, we have introduced the notion of *percentile accuracy* as a performance measure in building classification trees. This concept can more closely characterize a decision maker's risk profile in building a decision tree. We developed a GA (GAK) for constructing a tree based on this new measure, and conducted experiments with three different risk profiles. GAMin for $K = 0$, GAMedian for $K = 50$, and GAMax for $K = 100$.

We also investigated the effect of introducing diversity into our GA by allowing the algorithm to simultaneously focus on two distinct percentile criteria. We tested C4.5, GAMin, GAMedian, GAMax, and two variants of our bicriterion genetic algorithm (BGA and BGAC) on five data sets. Our results

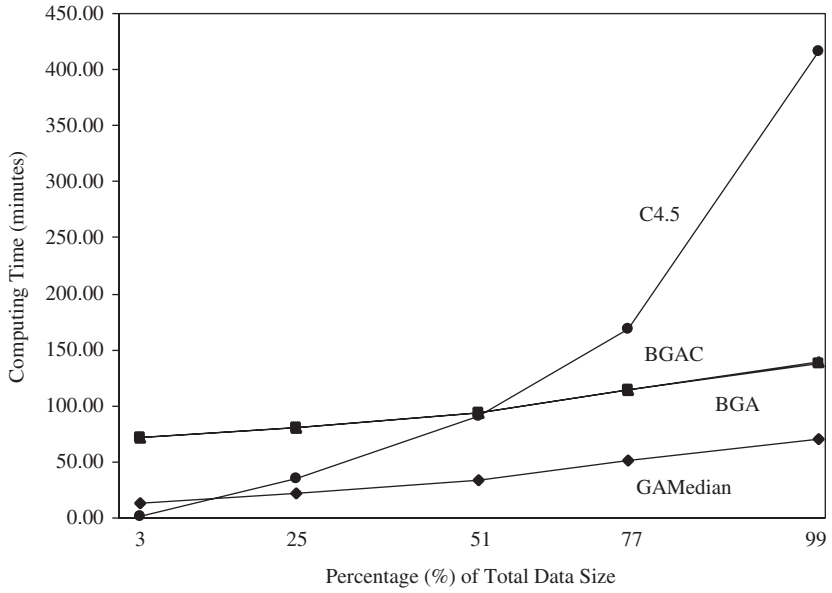

Fig. 4. Computation times (in minutes) of C4.5, GAMedian, BGA, and BGAC on different percentages of the transportation data set.

show that BGAC outperforms BGA, which in turn outperforms GAMin, GAMedian, and GAMax, which in turn outperform C4.5. We also demonstrate that GAMedian, BGA, and BGAC scale up well to handle very large data sets.

Our research underscores the usefulness of using GA approaches in generating decision trees, especially for very large data sets. It also demonstrates the value of incorporating diversity, via fitness criterion perturbation, into the GA to obtain better classification trees.

We believe the idea of introducing diversity within a GA, as envisaged in this paper has broader applications than the field of data mining. GAs are frequently used as heuristics for $\mathcal{NP}$-hard combinatorial optimization problems. We believe the idea of introducing genetic diversity, via fitness function perturbation as proposed in this paper and our earlier work, has the potential to improve the results of GAs when applied to a wide variety of $\mathcal{NP}$-hard combinatorial optimization problems.

References

[1] Breiman L, Friedman JH, Olshen RA, Stone CJ. Classification and regression trees. Monterey, CA: Wadsworth and Brooks/Cole; 1984.
[2] Quinlan JR. C4.5: programs for machine learning. San Mateo, CA: Morgan-Kaufmann; 1993.
[3] Hastie T, Tibshirani R, Friedman JH. The elements of statistical learning: data mining, inference, and prediction. New York: Springer; 2001.
[4] Li X-B, Sweigart J, Teng J, Donohue J, Thombs L. A dynamic programming based pruning method for decision trees. INFORMS Journal on Computing 2001;13(4):332–44.
[5] Fu Z, Golden B, Lele S, Raghavan S, Wasil E. A genetic algorithm-based approach for building accurate decision trees. INFORMS Journal on Computing 2003;15(1):3–22.

[6] Fu Z, Golden B, Lele S, Raghavan S, Wasil E. Genetically engineered decision trees: population diversity produces smarter trees. Operations Research 2003;51(6):894–907.
[7] Blake C, Merz C. UCI repository of machine learning databases, http://www.ics.uci.edu/~mlearn/mlrepository.html. 1998.
[8] Freitas AA. Data mining and knowledge discovery with evolutionary algorithms. Berlin: Springer; 2002.
[9] Freitas AA. A survey of evolutionary algorithms for data mining and knowledge discovery. In: Ghosh A, Tsutsui S, editors. Advances in evolutionary computation. Berlin: Springer; 2002. p. 819–45.
[10] Koza JR. Concept formation and decision tree induction using the genetic programming paradigm. In: Schwefel H-P, Maenner R, editors. Parallel problem solving from nature. Berlin: Springer; 1991. p. 124–8.
[11] Marmelstein R, Lamont G. Pattern classification using a hybrid genetic program-decision tree approach. In: Koza J, editor. Proceedings of the Third Annual Conference on Genetic Programming. San Francisco: Morgan Kaufmann; 1998.
[12] Folino G, Pizzuti C, Spezzano G. Genetic programming and simulated annealing: a hybrid method to evolve decision trees. In: Poli R, Banzhaf W, Langdon W, Miller J, Nordin P, Fogarty T, editors. Proceedings of the Third European Conference on Genetic Programming. Edinburgh, Scotland: Springer; 2000. p. 294–303.
[13] Bot M, Langdon W. Application of genetic programming to induction of linear classification trees. In: Poli R, Banzhaf W, Langdon W, Miller J, Nordin P, Fogarty T, editors. Proceedings of the Third European Conference on Genetic Programming. Edinburgh, Scotland: Springer; 2000. p. 247–58.
[14] Ryan M, Rayward-Smith V. The evolution of decision trees. In: Koza J, editor. Proceedings of the Third Annual Conference on Genetic Programming. San Francisco: Morgan Kaufmann; 1998. p. 350–8.
[15] Kennedy H, Chinniah C, Bradbeer P, Morss L. The construction and evaluation of decision trees: a comparison of evolutionary and concept learning methods. In: Corne D, Shapiro J, editors. Evolutionary computing, Lecture notes in computer science. Berlin: Springer; 1997. p. 147–61.
[16] Cantú-Paz E, Kamath C. Inducing oblique decision trees with evolutionary algorithms. IEEE Transactions on Evolutionary Computation 2003;7(1):54–68.
[17] Bauer E, Kohavi R. An empirical comparison of voting classfication algorithms: bagging, boosting and variants. Machine Learning 1999;36:105–39.
[18] Freund Y, Schapire R. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 1997;55:119–39.
[19] Breiman L. Arcing classifiers. Annals of Statistics 1998;24:801–49.
[20] Breiman L. Bagging predictors. Machine Learning, 1996;24:123–40.
[21] Lim T-S, Loh W-Y, Shih Y-S. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. Machine Learning 2000;40:203–29.
[22] Fu Z. Using genetic algorithms to develop intelligent decision trees. PhD dissertation, University of Maryland, College Park, MD: 2000.
[23] Berger JO. Statistical decision theory and Bayesian analysis. New York: Springer; 1985.
[24] Lehmann EL. Theory of point estimation. New York: Wiley; 1983.
[25] Lincoff GA. National Audubon Society field guide to North American mushrooms. New York: Alfred A Knopf; 1981.
[26] Maulin ML. Maintaining diversity in genetic search. In: Proceedings of the Fourth National Conference in Artificial Intelligence. Menlo Park, CA: AAAI Press; 1984. p. 247–50.
[27] Whitley D. The GENITOR algorithm and selection pressure: why rank-based allocation of reproductive trials is best. In: Proceedings of the Third International Conference of Genetic Algorithms. San Mateo, CA: Morgan Kaufmann Publishers; 1989. p. 116–21.
[28] Michalewicz Z. Genetic algorithms (+) data structures = evolution programs. New York: Springer; 1996.
[29] Gale D, Shapley LS. College admissions and the stability of marriage. American Mathematical Monthly 1962;69:9–15.