GENETICALLY ENGINEERED DECISION TREES: POPULATION DIVERSITY PRODUCES SMARTER TREES

ZHIWEI FU

Fannie Mae, 4000 Wisconsin Avenue NW, Washington, DC 20016, zhiwei_fu@fanniemae.com

BRUCE GOLDEN, SHREEVARDHAN LELE, and S. RAGHAVAN

Robert H. Smith School of Business, University of Maryland, College Park, Maryland 20742
bgolden@rhsmith.umd.edu • slele@rhsmith.umd.edu • raghavan@umd.edu

EDWARD WASIL

Kogod School of Business, American University, Washington, DC 20016, ewasil@american.edu

The second author dedicates this paper to the memory of his mother, Gloria D. Golden (1927–2003).

When considering a decision tree for the purpose of classification, accuracy is usually the sole performance measure used in the construction process. In this paper, we introduce the idea of combining a decision tree's expected value and variance in a new probabilistic measure for assessing the performance of a tree. We develop a genetic algorithm for constructing a tree using our new measure and conduct computational experiments that show the advantages of our approach. Further, we investigate the effect of introducing diversity into the population used by our genetic algorithm. We allow the genetic algorithm to simultaneously focus on two distinct probabilistic measures—one that is risk averse and one that is risk seeking. Our bivariate genetic algorithm for constructing a decision tree performs very well, scales up quite nicely to handle data sets with hundreds of thousands of points, and requires only a small percent of the data to generate a high-quality decision tree. We demonstrate the effectiveness of our algorithm on three large data sets.

(*Statistics, data analysis: data mining. Marketing, estimation/statistical techniques: decision trees. Computers/computer science, artificial intelligence: genetic algorithms.*)
Received February 2001; revisions received March 2002, October 2002; accepted December 2002.
Area of review: Computing and Decision Technology.

1. INTRODUCTION

Decision trees are a popular type of classification system (e.g., see Ripley 1996, Jain et al. 2000). When compared to other classification methods such as logistic regression, neural networks, and nearest neighbor rules, the chief attraction of decision trees is that they are easy to interpret in terms of the predictor variables that are used in constructing the tree. Algorithms for constructing decision trees vary according to the method employed for constructing the different nodes of the tree and according to the method used for collapsing unnecessary subtrees of the tree (this is called pruning the tree).

In Fu et al. (2003), we considered the development of decision trees using a genetic algorithm called GAIT (genetic algorithm for intelligent trees). It was seen that GAIT performed better than logistic regression and C4.5. In this paper, we consider two distinct and significant issues—one arising in the development of decision trees and the second arising in the development of genetic algorithms. We study these two issues in the context of decision trees generated using GAIT.

First, we treat the classification accuracy of a tree as a random variable. In contrast to most tree construction and pruning algorithms that focus implicitly and exclusively on the expected value of the classification accuracy, we consider the variance of the tree's accuracy as well. We use the expected value and variance properties of a tree by considering a new measure for assessing the performance of a tree. The proposed performance measure is the probability that the accuracy of the tree exceeds a prespecified threshold. It is shown that, depending on the value of the threshold that is specified, the probabilistic performance measure used for tree development can be described as either risk averse or risk seeking.

Second, we consider two distinct objectives in the fitness function of our genetic algorithm. Instead of focusing exclusively on optimizing a scalar performance measure (such as either the expected value of the accuracy or the probability of exceeding a threshold accuracy), we allow our genetic algorithm to be simultaneously mindful of two distinct probabilistic performance measures. We choose one of these measures to be risk averse while the other is chosen to be risk seeking. Because the genetic algorithm considers both of these measures, a diverse population of decision trees emerges. As a result, the genetic algorithm performs very well.

Our computational experiments are conducted on three large data sets. Initially, we focus on a real-life marketing data set obtained from a firm in the transportation services industry. This transportation marketing data set contains demographic and usage information for approximately 440,000 customers and 11 key variables.

The rest of this paper is organized as follows. In §2, we describe the probabilistic criterion for measuring the performance of a decision tree. An overview of the univariate GAIT algorithm that was developed in Fu et al. (2003) is provided in §3. In §4, we describe the results of our computational experiments using the probabilistic performance measure as the univariate fitness function in GAIT. In §5, we provide results on extending GAIT's fitness function to a bivariate function that keeps track of two distinct probabilistic performance measures. Scaling issues are addressed in §6 and two additional data sets are studied in §7. Conclusions are given in §8.

We compare our approach with that of C4.5. Lim et al. (2000) perform a comprehensive review and comparison of 33 classification algorithms (22 of these are decision tree algorithms, 9 are statistical algorithms, and 2 involve neural networks). The authors record prediction accuracy and training time over 32 data sets. They conclude that C4.5 has good classification accuracy and that it is, by far, the fastest of the 33 algorithms. Because our data sets are substantially larger that those studied by Lim et al. and a key focus of our work is computational effort, it seems sufficient to use C4.5 as a benchmark.

2. A PROBABILISTIC CRITERION FOR SCORING CLASSIFICATION TREES

Suppose we have a set of competing classification trees, each of which is intended to classify items from a certain population. We wish to assign a score to each tree that measures the goodness of the tree in classifying items from the given population. Such a score may be used to select the best tree out of the set of available trees. One method of assigning a score to each tree is to randomly select a subset of the population, called the scoring set, and record the overall classification accuracy of each tree on this common scoring set. Let n denote the size of the scoring set.

Now, the accuracy of a tree with respect to a randomly selected scoring set is actually a random variable. Its value depends on the data set selected for scoring. Different data sets, each of size n, may lead to different values of accuracy for a given tree. Because the accuracy with respect to a randomly selected data set is a random variable, it has an expected value and a variance. Most scoring procedures for classification trees focus on the expected value of the classification accuracy (e.g., see Berry and Linoff 1997) while ignoring the variance of the classification accuracy.

The variance of the accuracy of a tree measures the consistency of the tree in correctly classifying items. Consistency of accuracy is clearly a desirable property of a classification tree. Furthermore, the variance of accuracy has been linked to other desirable properties of a tree, such as simplicity and explainability. For example, Fu (2000) has shown that trees with smaller variances tend to have a smaller depth and a smaller number of nodes (i.e., they are less bushy). Ideally, we would like to select a tree that has a high mean accuracy while simultaneously having a low variance of accuracy.

To explicitly acknowledge the fact that the accuracy of a tree with respect to a randomly selected data set is a random variable, we consider the probability distribution of the accuracy. In particular, we consider the probability that the accuracy exceeds or falls below a preset threshold as a criterion function.

Suppose we use the probability that the accuracy exceeds a preset threshold as the criterion function to evaluate the goodness of a tree. Then, the tree that maximizes this criterion function is the best tree in the set of available trees. Let us call this criterion for determining the optimal tree Form 1. On the other hand, we may use the probability that the accuracy falls below a preset threshold as another criterion function for evaluating a tree. In this case, the tree that minimizes this criterion function is the best tree in the set of available trees. Let us call this criterion for determining the optimal tree Form 2.

Form 1 may be considered to be a risk-seeking criterion as it seeks to maximize the likelihood of a desirable outcome. Form 2 may be considered to be a risk-averse criterion as it seeks to minimize the likelihood of an undesirable outcome. For a fixed threshold, the two forms will lead to identical solutions because maximizing the probability of exceeding a threshold is equivalent to minimizing the probability of falling below that *same* threshold. The distinction between the two forms in the sense of being risk seeking and risk averse arises from the *value* of the threshold that is used in the criterion function. We address the issue of selecting the threshold value in greater detail below, and again in §4.

Let $T_i(n)$ denote the accuracy of the ith tree (out of the available set of trees) on a randomly selected scoring set of size n. We abbreviate this to T_i because n will be held fixed. Let μ_i and σ_i denote the mean and standard deviation of T_i. They can be estimated in two ways. Noting that T_i is the mean of n Bernoulli random variables, it follows that $\sigma_i = \sqrt{\mu_i(1-\mu_i)/n}$.

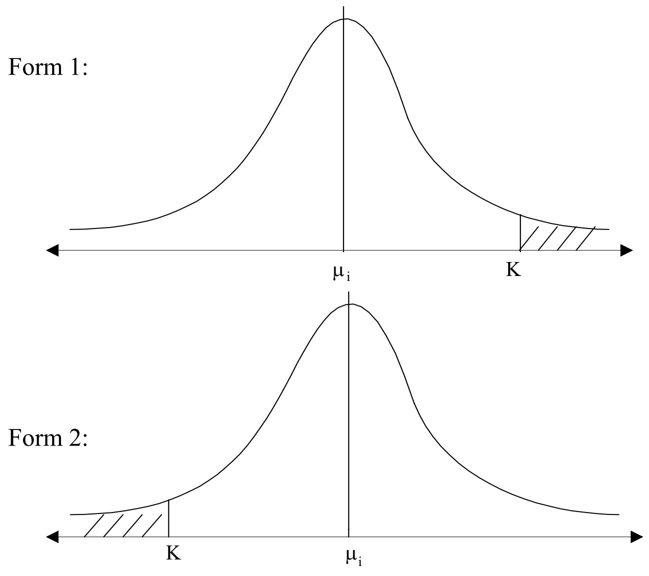
Let a_i denote the value of the accuracy of the tree on a single scoring set of n cases. Then, μ_i can be estimated by a_i, and σ_i can be estimated by $\sqrt{a_i(1-a_i)/n}$. However, this estimate of σ_i is extremely variable. To get a more accurate estimator of σ_i, the tree can be scored on m randomly selected data sets, each of size n. Let $\bar{a}_i$ and s_i denote the mean and standard deviation of the m sample accuracies. They are the estimates for μ_i and σ_i, respectively. In our study, we use this latter approach to estimate μ_i and σ_i.

Let K denote a preset threshold value for the accuracy T_i. Because T_i is a mean of n Bernoulli random variables, its distribution tends to the normal distribution when n is large. Then, the probability that T_i exceeds the threshold K, $\mathrm{P}\{T_i > K\}$ is $\mathrm{P}\{Z > (K-\mu_i)/\sigma_i\}$, where Z denotes a standard normal random variable. This probability can be estimated by plugging in the estimates for μ_i and σ_i, i.e., by $\mathrm{P}\{Z > (K-\bar{a}_i)/s_i\}$. Similarly, $\mathrm{P}\{T_i < K\}$ can be estimated by $\mathrm{P}\{Z < (K-\bar{a}_i)/s_i\}$.

Let z_i denote $(K - \bar{a}_i)/s_i$, the z-score of K with respect to the ith tree. Then, employing Form 1 of the criterion function, i.e., maximizing $\mathrm{P}\{T_i > K\}$, is equivalent to minimizing z_i (see Figure 1). Furthermore, employing Form 2 of the criterion function, i.e., minimizing $\mathrm{P}\{T_i < K\}$, is also equivalent to minimizing z_i. Therefore, the problem of determining the best tree using either Form 1 or Form 2 of the criterion function reduces to finding the z-score of each tree (with respect to threshold K) and then selecting the tree with the lowest z-score.

We return to the issue of selecting the value of the threshold, K. It is important to consider the value of the threshold in the context of the range of probable accuracies of the available trees. Let $a_{\min}$ and $a_{\max}$ denote the smallest and largest estimated mean accuracies in the set of available trees. A selection procedure that maximizes the probability that the accuracy of a tree exceeds $a_{\max}$ would be considered a risk-seeking selection procedure, as it attempts to maximize the probability of an extremely desirable outcome. Conversely, a selection procedure that minimizes the probability that the accuracy of a tree falls below $a_{\min}$ would be considered a risk-averse selection procedure, as it seeks to minimize the probability of an extremely undesirable outcome. By choosing an appropriate value of K between $a_{\min}$ and $a_{\max}$, one can control the degree of risk aversion of the tree-scoring procedure.

Figure 1. Risk-seeking (Form 1) and risk-averse (Form 2) criteria.

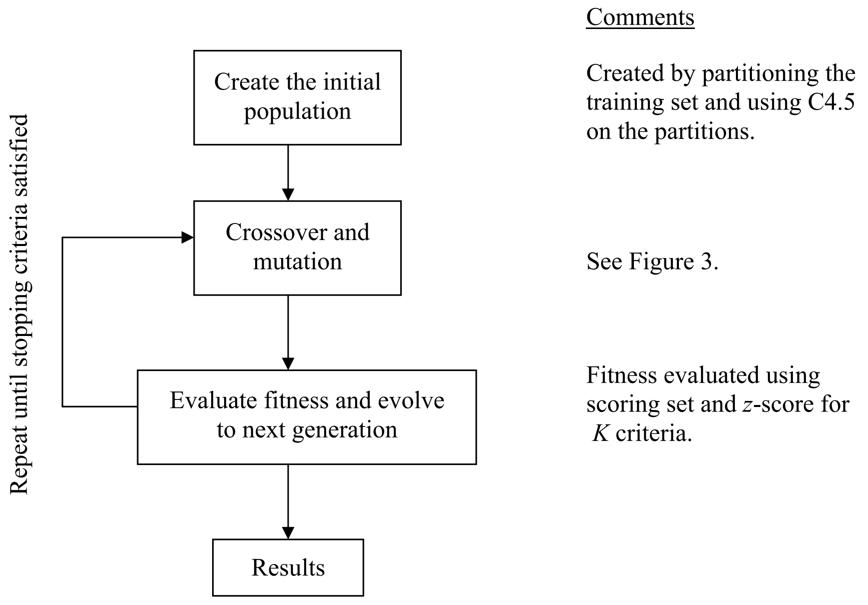


3. GENETIC ALGORITHM FOR INTELLIGENT TREES

In Figure 2, we provide an overview of GAIT—the genetic algorithm for classification trees that we developed in Fu et al. (2003). The algorithm starts with an initial population of classification trees (referred to as trees hereafter). These are the set of trees (parents) at the start of the first generation. GAIT then applies the typical genetic operations of crossover and mutation to generate new trees (children). It then evaluates the fitness of all trees in a generation to determine the trees, among both the parents and children, that survive the generation. The trees that survive a generation are taken to the next generation, and the remaining trees are deleted. This procedure is repeated for a specified number of generations, or until some specified level of convergence is reached. We now elaborate on some of the details of GAIT.

Initial Population of Trees. Our procedure splits the data set available for tree generation purposes into two parts: a *training set* and a *scoring set*. The training set is used to generate the initial set of trees as follows.

Figure 2. GAIT overview.

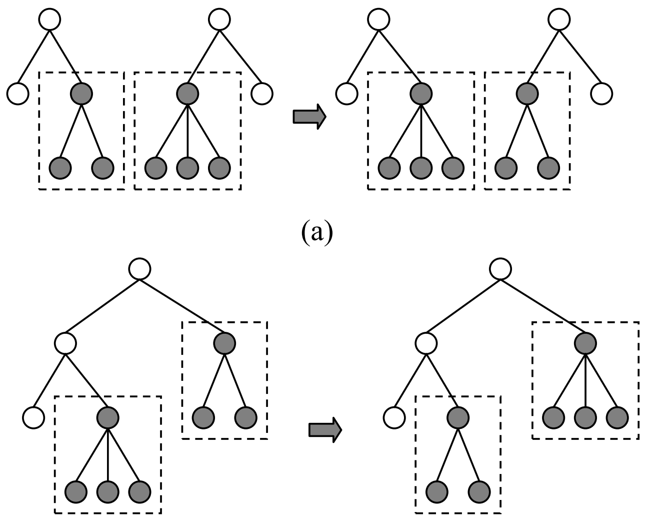

Comments
Create the initial population
Created by partitioning the training set and using C4.5 on the partitions.
Crossover and mutation
See Figure 3.
Evaluate fitness and evolve to next generation
Fitness evaluated using scoring set and z-score for K criteria.
Repeat until stopping criteria satisfied
Results

If the training set consists of N_t points, then GAIT generates the initial set of, say k, trees by randomly sampling N_t/k points, either with or without replacement from the training set. (N_t is chosen to be an integer multiple of k.) In this paper, we sample with replacement from the training set. A tree is generated using the popular C4.5 algorithm (see Quinlan 1993) on this sample of N_t/k points. The process continues until k trees are generated.

Crossover and Mutation. The crossover operation takes two parent trees to generate two new children trees. It randomly selects a node in each parent and exchanges the subtrees rooted at these nodes to obtain two children trees. Roulette wheel parent selection ensures that the best (most fit) current trees are given more reproductive chances than other current trees (see Michalewicz 1996 for details). Crossover is accepted with a probability equal to the crossover rate. Following crossover, a mutation operator is applied and then accepted with a probability equal to the mutation rate. Mutation involves the exchange of subtrees *within* a tree. Two nodes are selected at random within the tree and the subtrees rooted at each of these nodes are exchanged. We make sure that neither of these nodes is in the other node's subtree. In Figure 3, we show the crossover and mutation operations.

Fitness Function. In a genetic algorithm, with an elitist strategy, we score the population of feasible solutions on a fitness criterion to determine the solutions that survive from one generation to the next. In our application of GAIT (see Fu et al. 2003), we use the classification accuracy of a tree on the scoring set as the fitness criterion. We modify the fitness function to the probabilistic criterion described in §2. To be specific, we divide the scoring set of $N_s = nm$ points into m sets of n data points each. We determine the classification accuracy of the tree on each of these m sets, and compute the average classification accuracy and the standard deviation of the classification accuracy of the classification tree on the scoring set. We then use this to obtain the z-score of the tree for the particular choice of the K criterion. Recall that a tree with a lower z-score represents a higher fitness value. In GAIT, the number of trees that survive from one generation to the next is set equal to the number of initial trees (i.e., k).

Figure 3. (a) Crossover operation. (b) Mutation operation.



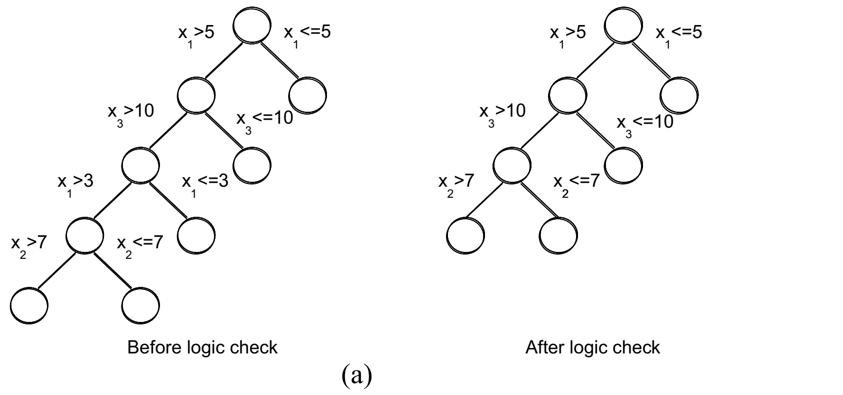
Logic Checks and Pruning. It is possible for the crossover and mutation operations to result in trees that have logically inconsistent rules. This does not actually affect the classification accuracy of a tree because logically inconsistent rules will never classify any data point (because no data point will satisfy the logically inconsistent conditions of the rule). However, it is advantageous to prune the tree by deleting the branches corresponding to these logically inconsistent rules. Consequently, within the genetic algorithm, we check and correct these logical inconsistencies in the trees (see Figure 4). Notice that the logic checks could be conducted either immediately after a crossover operation or a mutation operation, at the end of a generation for just the trees that will survive, or at the end of GAIT (i.e., ignoring logic violations until the end of the algorithm). In Fu et al. (2003), we found it advantageous to perform the logic checks immediately after a crossover or a mutation, or at the end of a generation. These logic checks improve the accuracy, reduce the space requirements, and add very little to the running time. In terms of the trees generated, there is no difference between performing the logic checks immediately after a crossover or a mutation, or at the end of a generation. The two are equivalent, but it is computationally more efficient to delay the logic checks to the end of a generation. We also employ the following standard pruning criteria to increase the ability of the tree to classify data sets other than the data set on which it is generated. Pruning starts from the bottom (leaf) 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 prespecified value (e.g., 0.5%).

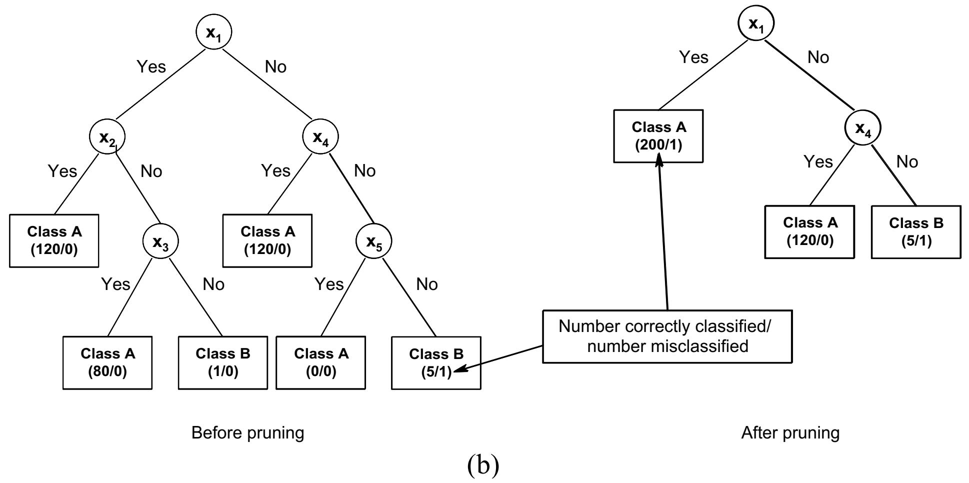
In §4, we will perform computational experiments using GAIT with the new fitness-scoring function on the transportation marketing data set. Because the fitness function depends on the K value selected, for convenience we will denote the genetic algorithm with the new classification function by GAK. In our experiments, we will use multiple K values to assess the behavior of the genetic algorithm. Consequently, we use the term GA to refer to the set of genetic algorithms with different K values, and GAK (with K replaced by the appropriate value) to refer to the genetic algorithm with a particular choice of the K value.

4. COMPUTATIONAL EXPERIMENTS WITH THE GENETIC ALGORITHM

We apply our genetic algorithm with three different K values to the transportation marketing data set. Each customer

Figure 4. (a) Logic check. (b) Pruning.





record in this data set contains demographic and usage information. Furthermore, each customer is identified on the basis of whether or not the customer reuses the firm's services in a certain window of time following a marketing promotion. It is of considerable interest to the firm to identify patterns among repeat customers. Recognizing such patterns is beneficial not only for developing a focused marketing strategy, but also for evaluating existing marketing efforts.

4.1. Experimental Design

From the entire set of 440,000 points in the transportation marketing data set, we randomly select a training set of $N_t = 10{,}000$ points, a scoring set of $N_s = 4{,}000$ points, and a test set of 4,000 points. We set the test set aside and do not use it to train or score decision trees.

We randomly sample with replacement from the training set to generate $k = 200$ unique decision trees. We partition the scoring set into $m = 40$ subsets of $n = 100$ points each and calculate the accuracy of each of the 200 decision trees on each subset.

We then compute the average accuracy ($\bar{a}_i$) and standard deviation (s_i) of each decision tree on the 40 scoring subsets and calculate the z-score of the ith tree with respect to a threshold K; that is, $z_i = (K - \bar{a}_i)/s_i$. (The choice of K is discussed below.) We sort the z-scores of the 200 decision trees and select the 50 trees with the lowest z-scores to form the initial population for our genetic algorithm.

We run our genetic algorithm for 50 generations with a crossover rate of 1.00 and a mutation rate of 0.01 and save the 50 best trees of each generation. We perform 10 replications that result in 10 best final trees. 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.

There are three choices for the threshold value K: L which corresponds to the risk-averse criterion (Form 2), U which corresponds to the risk-seeking criterion (Form 1), and F which is a value between L and U. Thus, in the fitness function of our genetic algorithm, the value of K is set to L, U, or F.

We set the values of L, U, and F based on observations made in conducting preliminary experiments with our genetic algorithm on a subset of the entire data set. We set $\text{L} = 0.5500$, which is slightly below the smallest classification accuracy that we observed, and $\text{U} = 0.8200$, which is slightly above the largest classification accuracy that we observed in our preliminary experiments. We set $\text{F} = 0.7780$, which is the proportion of the majority class (non-reusers) in the training set.

Finally, we denote our genetic algorithm run with the three different values of K in the fitness function as GAL, GAU, and GAF. In the case of GAL, the fitness function

Table 1. Computational results for the genetic algorithm with different values of K on the transportation marketing data set.

	GAL	GAU	GAF
Criterion	L	U	F
Accuracy	0.7770	0.7869	0.7871
Standard error	0.051	0.056	0.055
Depth	3.4	3.1	3.7
Number of nodes	6.0	5.9	7.4
z-score	−4.475	0.597	−0.161
Time (minutes)	14.76	14.56	13.21

Note. Average results are reported on the test sets for the best final tree that the algorithm generated.

Table 2. Computational results for C4.5 on the transportation marketing data set.

	C4.5		
Criterion	L	U	F
Accuracy	0.7680	0.7680	0.7680
Standard error	0.082	0.082	0.082
Depth	5.0	5.0	5.0
Number of nodes	9.0	9.0	9.0
z-score	−2.672	0.637	0.123
Time (minutes)	1.08	1.08	1.08

Note. C4.5 is run on the 14,000 point (training + scoring) set. Average results are reported on the test sets for the final C4.5 tree evaluated on three different criteria.

for the ith tree is $z_i = (\mathrm{L} - \bar{a}_i)/s_i$; for GAU, it is $z_i = (\mathrm{U} - \bar{a}_i)/s_i$; and for GAF, it is $z_i = (\mathrm{F} - \bar{a}_i)/s_i$. Recall that, to determine the best tree in all three cases (algorithms), we find the z-score of each tree (with respect to threshold $K = \mathrm{L}$, U, or F) and then select the tree with the lowest z-score.

4.2. Computational Results

In Table 1, we report the computational results on the 40 test subsets for the genetic algorithm with three different values of the threshold K. We report results for GAL with a threshold value of L; for GAU with a threshold value of U; and for GAF with a threshold value of F.

The results in Table 1 are averages on the 40 test subsets over 10 replications. For example, we ran the GAF algorithm, generated a best final tree, evaluated the tree on each of the 40 test subsets, and calculated average summary measures. We repeated this 10 times and then averaged over all 10 replications. In the case of GAF, the average accuracy of all 10 runs is 0.7871.

In Table 2, for comparison purposes, we report results for C4.5. We ran C4.5 on 14,000 points (10,000 training + 4,000 scoring), and evaluated the single generated tree on the 40 test subsets. We point out that all of the summary measures except the z-score are the same because there is only one tree that emerges from C4.5. Using the three values for K (that is, $\mathrm{L} = 0.5500$, $\mathrm{U} = 0.8200$, and $\mathrm{F} = 0.7780$), we compute the z-scores for C4.5 and show them in the penultimate row of the table. Observe that the time for one run of C4.5 is much less than the average time for one run of the genetic algorithm.

In Table 3, we provide the results of a paired-difference t-test for the improvement of GAL, GAU, and GAF over C4.5 on the test sets when C4.5 is run on 14,000 points. We see that GAL, GAU, and GAF are significantly more accurate than C4.5 (p-values $\leqslant 0.0018$) and generate z-scores that are significantly lower than C4.5 (p-values $\leqslant 0.0753$).

Based on our computational study, we conclude that our genetic algorithm outperforms C4.5 in terms of accuracy and z-score. Furthermore, GAF (genetic algorithm with threshold value F) generates trees that are, on average, more accurate than GAL and GAU.

In the next section, we try to address the problem of how to set the "correct" value of K by developing a genetic algorithm that considers two threshold values at the same time.

5. THE BENEFITS OF DIVERSITY

The open literature frequently espouses the benefits of population diversity within a genetic algorithm (e.g., see Maulin 1984; Whitley 1989; Michalewicz 1996, p. 58). However, it is often difficult to quantify or measure the benefits of introducing such diversity into genetic algorithms. There appear to be several reasons for this. We highlight the two main ones. First, it appears that diversity is introduced in an ad-hoc manner in most genetic algorithms. Thus, it becomes difficult to quantify the improvements in a genetic algorithm due to the introduction of diversity.

Table 3. Results of a paired-difference t-test for the improvement of GAL, GAU, and GAF over C4.5 on the test sets from the transportation marketing data set.

	GAL		GAU		GAF	
Comparison to	C4.5		C4.5		C4.5	
Criterion	L		U		F	
	t-statistic	p-value	t-statistic	p-value	t-statistic	p-value
Accuracy	3.9130	0.0018	15.4375	0.0000	15.1579	0.0000
Standard error	−18.5698	0.0000	−36.7496	0.0000	−15.4557	0.0000
Depth	−7.2	0.0000	−6.9	0.0000	−2.5	0.0166
Number of nodes	−7.1	0.0000	−3.7	0.0027	−1.0	0.1756
z-score	−15.1011	0.0000	−1.5711	0.0753	−13.5334	0.0000

Second, there appear to be few controlled experiments to validate the benefits of diversity. We address both of these issues in our investigation of diversity. To begin, we motivate the methodology used to introduce diversity by means of a simple graphical analog.

Suppose we represent any classification tree as a point on a two-dimensional graph, where the x-axis represents the standard deviation, σ, and the y-axis measures the mean classification accuracy of the tree, μ. The probabilistic fitness function has a very simple interpretation here. Notice that the z-score of a tree i, which we denote as z_i, is equal to $(K - \mu_i)/\sigma_i$. Rearranging terms gives the equation of a line $\mu_i = -z_i\sigma_i + K$. Recall that for any particular K value, we wish to find the tree with the lowest z-score. That corresponds to finding the line with maximum slope $(-z_i)$ and specified intercept K, with the restriction that the line pass through a point (σ_i, μ_i) associated with a classification tree. We illustrate this concept in Figure 5.

We observe that classification trees are actually generated within the genetic algorithm. Therefore, the genetic algorithm via its crossover and mutation operations, and its fitness function and elitist strategy, generates points in this two-dimensional representation. It also finds the point (among the points it generates) that maximizes the slope of the line with specified intercept K.

In this two-dimensional representation, concentrating on the U criterion (when appropriately set) is equivalent to trying to discover points in the northeast corner of the universe of points. The L criterion is equivalent to focusing on the northwest corner of the universe of points. The F criterion is equivalent to focusing on the northern part of the universe of points. In other words, each criterion tries to discover and construct classification trees in different parts of this universe.

To introduce diversity into this process, we consider the effect of simultaneously relying on two K criteria (L and U) that are, for the moment, different from our final K criterion (F). The idea is to examine whether using fitness functions that are designed to discover points that are to the left and right of our desired objective combined with the operations of the GA can find better trees with respect to our desired objective (i.e., scored with respect to the F criterion).

With this motivation in hand, we now describe an adaptation of our GA to handle two fitness criteria.

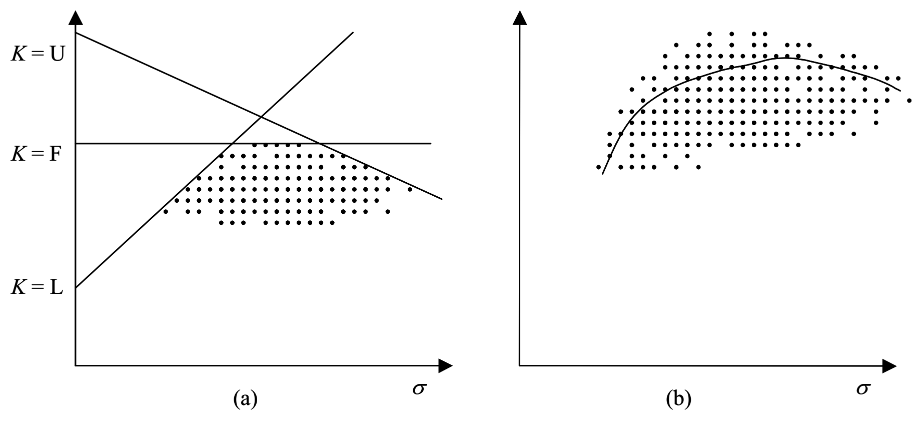
5.1. Bivariate Genetic Algorithm

The bivariate genetic algorithm (BGA) is a simple adaptation of the basic GAIT algorithm. The main difference between GAIT and BGA is in determining the trees that survive and are carried over to the next generation. We elaborate further on this distinction.

At each generation, BGA maintains two lists—an L list consisting of trees ranked by the L criterion (in increasing order of z-scores) and a U list consisting of trees ranked by the U criterion. Suppose the number of trees that survive a generation is N. Then, the length of each list is N. Notice that the number of distinct trees in these two lists is at least N and no more than $2N$. For the moment, assume that the trees on the L and U lists are distinct. In this case, we choose the N trees that survive by selecting the top $N/2$ trees on the L list and the top $N/2$ trees on the U list.

To deal with the situation where the trees on the two lists are not distinct, we modify the selection algorithm as follows. Let SU and SL denote the ranked lists of trees that survive the generation. At the start of each generation both of these lists are empty and the maximum size of both lists is $N/2$ trees. We select the first item on the U list and add it, in ranked order, to the SU list if the tree is neither on the SU list nor on the SL list, and the SU list is not full. Next, we consider the first item on the L list and add it, in ranked order, to the SL list if the tree is neither on the SL list nor on the SU list, and the SL list is not full. This procedure continues by considering the second item on the U list and the second item on the L list, the third item on

Figure 5. (a) For any particular K value we wish to find the line with maximum slope and specified intercept K that passes through a point in this graph. Observe all points (trees) lie below the lines. (b) Will diversity help us discover better trees (points)? Points below the curve are those discovered by GA using different K values. Points above are what we hope to obtain by simultaneously using two K criteria.



the U list and the third item on the L list, etc., until both the SU list and the SL list are full. At that point, we have $N/2$ trees on the SU list and $N/2$ trees on the SL list for a total of N distinct trees that survive the generation. Notice that this approach is identical to the stable marriage algorithm (see Gale and Shapley 1962). We took this approach to ensure that we have N trees that survive the generation. We want to have a fair representation of the U lists choices for the $N/2$ slots it commands and a fair representation of the L lists choices for the $N/2$ slots it commands.

In the last generation of BGA, we generate as output the best tree as scored by the U criterion, as scored by the L criterion and, because we wish to investigate the benefits of diversity, as scored by the F criterion.

We compared BGA to GAL, GAU, and GAF by conducting an identical set of experiments to those described in §4. Recall that N was 50 in these experiments (that is, 50 trees survived a generation). We summarize these results in Tables 4 and 5.

The results in Table 4 are averages on the 40 test sets over 10 replications. In other words, we ran BGA and, at the end of the algorithm, obtained three trees: the best tree with respect to the L criterion, the best tree with respect to the U criterion, and the best tree with respect to the F criterion. We evaluated each of these trees on the test set to calculate average summary measures for each tree. We repeated this procedure 10 times and averaged over the 10 replications to obtain the summary measures reported in Table 4. The average computation time of BGA was 71.05 minutes, which was significantly greater than GAL, GAU, and GAF.

The results in Table 4, as compared with the results reported in Table 1, are quite interesting. They show that BGA(L, U) generates more accurate trees, with smaller or equal standard error, smaller depth, fewer nodes, and lower z-score for all three scoring criteria (L, U, and F)! Given that the algorithm does not consider the F criterion except in the last generation, this shows that diversity has enabled this discovery of better trees than GAL, GAU, and GAF. Furthermore, we observe that diversity has also benefited the L and U criteria in the sense that we have obtained better trees with respect to the L criterion, when we introduced diversity by keeping good U trees in the population.

To test the significance of these improvements, we conducted a paired-difference t-test between the results of BGA and GAL, GAU, and GAF, respectively. These results are shown in Table 5. The results show that BGA generates significantly more accurate trees than GAL, GAU, and GAF (p-values $\leqslant 0.0105$) and generates z-scores that are significantly lower than GAL, GAU, and GAF (p-values $\leqslant 0.004$). The results also show that BGA generates leaner trees (as measured by depth and number of nodes) than GAL, GAU, and GAF.

Table 4. Computational results for the bivariate genetic algorithm using L and U as the two K criteria on the transportation marketing data set.

	BGA(L, U)		
Criterion	L	U	F
Accuracy	0.7790	0.7889	0.7890
Standard error	0.051	0.055	0.053
Depth	3.0	3.0	3.2
Number of nodes	5.4	5.7	5.6
z-score	−4.531	0.566	−0.211

Note. Average results are reported on the test sets for the best final tree that the algorithm generated for the L, U, and F criteria. Average running time of BGA is 71.05 minutes.

Although the running time of BGA is significantly greater than GA, one advantage of BGA over GA occurs when the decision maker is unsure about the choice of the value for K. In this situation, the decision maker could run BGA choosing a low K value (L) and a high K value (U). At the end of the last generation, by saving all of the trees in that generation, the decision maker can experiment with many different K values to test the various qualities of the trees (accuracy, depth, etc.). Unlike GA, the decision maker would not have to rerun the algorithm, because, as we have seen, the diversity in the population generates good trees for K values that lie between L and U. After evaluating the trees generated with different values for K, the decision maker may be in a better position to make a choice of the final K criterion (F).

5.2. Bivariate Genetic Algorithm with Convergence

The results of the previous section raise a natural question: Can guiding the BGA to a final K criterion provide even better classification trees? To answer this question, we

Table 5. Results of a paired-difference t-test for the improvement of BGA over GAL, GAU, and GAF on the test sets from the transportation marketing data set.

	BGA(L, U)					
Comparison to	GAL		GAU		GAF	
Criterion	L		U		F	
	t-statistic	p-value	t-statistic	p-value	t-statistic	p-value
Accuracy	2.7894	0.0105	4.4721	0.0008	4.3846	0.0009
Standard error	−1.5884	0.0733	−2.8809	0.0091	−2.3911	0.0202
Depth	−0.4	0.0186	−0.8944	0.1972	−1.6270	0.0691
Number of nodes	−2.5100	0.0167	−0.8944	0.1972	−1.5486	0.0779
z-score	−4.5879	0.0007	−3.3960	0.0040	−5.5062	0.0002

modify BGA to obtain a bivariate genetic algorithm with convergence (BGAC) as follows.

We are given three K values—a starting L value, a starting U value, and a final F value that lies between L and U. We want to obtain good classification trees as scored by the F criterion. Each generation of BGAC is identical to BGA, except that we modify the L and U values at each generation, increasing L and decreasing U, so that they converge to F at the final generation of BGAC. In particular, the increase in L at each generation is given by $(\mathrm{F}-\mathrm{L})$/(number of generations), and the decrease in U at each generation is given by $(\mathrm{U}-\mathrm{F})$/(number of generations). In the final generation, both U and L are equal to F, and the best trees on both the U list and the L list are the best trees with respect to the F criterion.

We ran an identical set of experiments on BGAC, as we did on BGA. Recall that the number of generations is 50. In the first column of Table 6, we report these results. The results show that the trees generated by BGAC are more accurate, have smaller standard error, smaller depth, smaller number of nodes, and lower z-score than BGA (using the F criterion) and GAF. This indicates that BGAC benefits from both diversity and the guidance to the final F criterion. BGAC's average computation time (70.97 minutes) is comparable to BGA's time (71.05 minutes).

To test the validity of our conclusions, we ran a paired-difference t-test between the results of BGAC and BGA. The t-test results are reported in the second and third columns of Table 6 and are with respect to the F criterion. We see that BGAC generates significantly more accurate trees than BGA (p-value $= 0.0052$) and trees with lower z-scores (p-value $= 0.0047$). The results also validate the conclusion that BGAC generates trees with smaller depth and smaller number of nodes.

Based on our computational study, we conclude that, except for computation time, BGAC outperforms BGA (i.e., BGAC generates trees that are of greater accuracy, smaller depth, smaller number of nodes, and lower z-score than BGA), which in turn outperforms GAF, which in turn outperforms C4.5. A tree with smaller depth and fewer nodes has a simpler set of rules. Consequently, with each succeeding algorithm, we are obtaining more accurate trees, with simpler rules and smaller variability.

For the current experiment, the computation times of BGAC and BGA are comparable, GA is faster than BGA, and C4.5 is faster than GA. However, before we draw any further conclusions about the computation time, we note that it is important to see how the algorithms scale. In the next section, we compare how the different genetic algorithms scale to handle larger training sets and scoring sets.

Table 6. Computational results for BGAC and paired-difference t-test results for the improvement of BGAC over BGA on the transportation marketing data set.

	BGAC(L, U, F)		
	F criterion	t-statistic	p-value
Accuracy	0.7923	3.2273	0.0052
Standard error	0.052	−2.3658	0.0211
Depth	2.7	−2.2361	0.0186
Number of nodes	4.8	−1.4446	0.0912
z-score	−0.274	−3.2925	0.0047

Note. Average results are reported on the test sets for the best final tree that the algorithm generated with $\mathrm{L} = 0.5500$, $\mathrm{U} = 0.8200$, and $\mathrm{F} = 0.7780$. Average running time of BGAC is 70.97 minutes. The z-score is with respect to the F criterion and the t-test is for the improvement on the test sets.

6. SCALABILITY EXPERIMENTS

In this section, we increase the size of the training set and the size of the scoring set for C4.5 and the three algorithms (GAF, BGA, and BGAC performed well in the previous experiments) to investigate issues of scalability. As before, all 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, we use the transportation marketing data set of approximately 440,000 data points and randomly select 4,000 points that are set aside to form a test set. We then generate a series of training and scoring combinations to assess scalability, as shown in the first two columns of Table 7. 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 4,000 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, and so on. In the same way, we partition the test set of 4,000 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 greater 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 z-score, classification accuracy, and standard error on the test set and computing time on the training and scoring sets for C4.5, GAF, BGA, and BGAC. With respect to C4.5, we use the training set *and* scoring set data as input and compute the accuracy on the test set. Note that although the sizes of the training, scoring, and test sets for the experiment with 3% of the data are the same as in §§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 4,000 points.

Table 7. Computational results for scaling experiments on the transportation marketing data set.

Size			C4.5				GAF			
Training (N_t)	Scoring (N_s)	%	Accuracy	Standard Error	*z*-score	Time (min)	Accuracy	Standard Error	*z*-score	Time (min)
10,000	4,000	3	0.7730	0.0719	0.0695	1.40	0.7877	0.0539	−0.1805	14.74
80,000	32,000	25	0.7820	0.0621	−0.0651	36.22	0.7939	0.0508	−0.3148	24.28
160,000	64,000	51	0.7870	0.0582	−0.1542	93.67	0.7949	0.0499	−0.3378	33.91
240,000	96,000	72	0.7880	0.0568	−0.1768	176.82	0.7952	0.0494	−0.3495	51.25
310,000	124,000	99	0.7880	0.0563	−0.1982	432.55	0.7954	0.0492	−0.3566	77.46

Size			BGA				BGAC			
Training (N_t)	Scoring (N_s)	%	Accuracy	Standard Error	*z*-score	Time (min)	Accuracy	Standard Error	*z*-score	Time (min)
10,000	4,000	3	0.7895	0.0520	−0.2234	70.31	0.7926	0.0513	−0.2846	70.53
80,000	32,000	25	0.7950	0.0494	−0.3449	78.38	0.7962	0.0488	−0.3735	81.31
160,000	64,000	51	0.7957	0.0483	−0.3690	91.49	0.7972	0.0474	−0.4049	92.75
240,000	96,000	72	0.7962	0.0480	−0.3792	108.32	0.7974	0.0467	−0.4144	11.17
310,000	124,000	99	0.7963	0.0475	−0.3885	128.12	0.7977	0.0467	−0.4211	132.36

Note. Results for five training and scoring combinations are based on the average of 10 replications for C4.5, GAF, BGA, and BGAC.

6.2. Computational Results and Analysis

In Table 7, we present the average accuracy, standard error, *z*-score, and running-time results for C4.5, GAF, BGA, and BGAC. We observe that the *z*-scores for all four algorithms decrease and, for the most part, accuracy increases as the size of the training and scoring combination increases. In addition, BGAC outperforms BGA, which outperforms GAF, which outperforms C4.5 with respect to both accuracy and *z*-score.

We also see that the computation time increases when the size of the training and scoring combination increases for C4.5, GAF, BGA, and BGAC. For the most part, we see that GAF takes the least amount of computation time in training and scoring. BGA and BGAC take nearly the same amount of computation time and require more computation time than GAF. C4.5 has, by far, the fastest growth rate in computation time. Its computation time is less than that of BGA and BGAC when the size of the training and scoring combination is small (up to 51%), but its computation time increases greatly thereafter. For example, when 99% of the total data set is used, C4.5 takes more than three times as long as BGAC.

To better illustrate the differences in performance, Figure 6 plots the classification accuracy of each algorithm (on an enlarged scale) against the size of the training and scoring combination as a percentage of the size of the

Figure 6. Classification accuracy of C4.5, GAF, BGA, and BGAC on five training and scoring combinations from the transportation marketing data set.

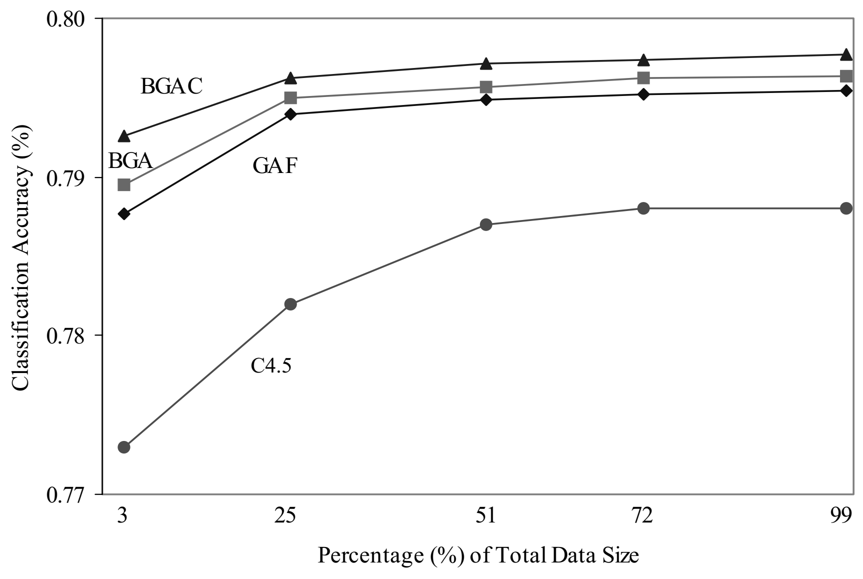


Table 8. Computational results for C4.5, GAL, GAU, and GAF on the adult data set.

	C4.5			GAL	GAU	GAF
Criterion	L	U	F	L	U	F
Accuracy	0.7980	0.7980	0.7980	0.8010	0.8039	0.8070
Standard error	0.0680	0.0680	0.0680	0.0390	0.0410	0.0401
Depth	5.0	5.0	5.0	3.6	3.6	3.5
Number of nodes	11.0	11.0	11.0	7.5	7.3	7.4
z-score	−2.9098	0.6173	−0.6731	−5.1529	0.8780	−1.3659
Time (minutes)	1.34	1.34	1.34	17.05	16.98	17.12

Note. L = 0.6000, U = 0.8400, and F = 0.7522.

data. We see that GAF, BGA, and BGAC clearly outperform C4.5. We also observe that classification accuracy jumps most dramatically when the training and scoring size increases from 3% to 25%. Beyond that point, classification accuracy levels off. It is important to remark that, using only 3% of the data, BGA and BGAC outperform C4.5 using 99% of the data. Furthermore, GAF, using 3% of the data, is comparable to C4.5 using 99% of the data. As noted earlier, BGAC outperforms BGA, which outperforms GAF.

We make two observations from these experiments. First, GAF, 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, GAF, BGA, and especially BGAC, require only a small percent of the data to generate high-quality decision trees.

7. ADDITIONAL COMPUTATIONAL EXPERIMENTS

We analyzed two other large real-life data sets. One was the "adult" data set that has become well known in the data-mining literature; the other data set was obtained from a direct marketing campaign for a financial product.

7.1. Adult Data Set

The adult data set can be obtained from the UCI Machine Learning Repository (see Blake and Merz 1998). 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 more than \$50,000). The proportion of the majority class (low income) in the data set is 75.22%. For this data set, we set L = 0.6000 (which is approximately the lowest classification accuracy observed in preliminary experiments), U = 0.8400 (which is approximately the highest accuracy observed), and F = 0.7522 (which is the proportion of the majority class, or the accuracy of the naïve rule that predicts the majority class for all observations).

In Tables 8, 9, and 10, we provide the analysis for this data set. Table 8 presents results for C4.5, GAL, GAU, and GAF. The columns corresponding to C4.5 are similar to Table 2, while those corresponding to GAL, GAU, and GAF are similar to Table 1. Recall that for C4.5, all of the summary measures except the z-score are the same because there is only one tree that emerges from C4.5. Table 9 presents the results for BGA and BGAC. The columns corresponding to BGA are similar to Table 4, while the column corresponding to BGAC is similar to the first column of Table 6. Recall that with a single run of BGA, we obtain three trees (one for the L, U, and F criteria, respectively). Consequently, the running times for BGA(L, U) under the various criteria in Table 9 are identical.

The results for this data set corroborate our findings with the transportation marketing data set. For example, in Table 8 we see that GAL, GAU, and GAF perform significantly better than C4.5. Furthermore, in Table 9 we see that the bivariate version BGA(L, U) performs significantly better than each of the univariate algorithms and that BGAC(L, U, F) performs significantly better than BGA(L, U). We also conducted paired-difference t-tests for the improvements of BGAC over BGA; BGA over GAL, GAU, and GAF; and GAL, GAU, and GAF over C4.5. We found that improvements in accuracy and z-score were statistically significant, with p-values of 0.02 or less, while the improvements in standard error, depth, and number of nodes were statistically significant with p-values of 0.09 or less. The scalability experiments summarized in Table 10 reveal a pattern that is similar to that found for the transportation marketing data.

7.2. Financial Services Data Set

This data set was obtained from a direct marketing campaign at a major financial services firm. This data set con-

Table 9. Computational results for BGA and BGAC on the adult data set.

	BGA(L, U)			BGAC(L, U, F)
Criterion	L	U	F	F
Accuracy	0.8099	0.8110	0.8119	0.8190
Standard error	0.0390	0.0400	0.0390	0.0340
Depth	3.3	3.1	3.1	2.6
Number of nodes	6.7	6.8	7.0	5.2
z-score	−5.3837	0.7247	−1.5328	−1.9646
Time (minutes)	23.21	23.21	23.21	22.39

Note. L = 0.6000, U = 0.8400, and F = 0.7522.

Table 10. Computational results for scaling experiments on the adult data set.

Size			C4.5				GAF			
Training (N_t)	Scoring (N_s)	%	Accuracy	Standard Error	z-score	Time (min)	Accuracy	Standard Error	z-score	Time (min)
10,000	4,000	32	0.7980	0.0670	−0.6836	1.84	0.8070	0.0400	−1.3700	16.85
15,000	6,000	48	0.8100	0.0620	−0.9323	15.20	0.8260	0.0380	−1.9432	20.38
20,000	8,000	64	0.8180	0.0572	−1.1503	31.77	0.8299	0.0370	−2.1027	24.10
25,000	10,000	80	0.8210	0.0560	−1.2286	60.22	0.8310	0.0370	−2.1297	31.25
30,000	12,000	95	0.8220	0.0560	−1.2464	104.53	0.8329	0.0370	−2.1834	40.41

Size			BGA				BGAC			
Training (N_t)	Scoring (N_s)	%	Accuracy	Standard Error	z-score	Time (min)	Accuracy	Standard Error	z-score	Time (min)
10,000	4,000	32	0.8119	0.0390	−1.5335	22.98	0.8189	0.0340	−1.9649	21.92
15,000	6,000	48	0.8320	0.0380	−2.1000	26.38	0.8360	0.0340	−2.4649	27.31
20,000	8,000	64	0.8339	0.0370	−2.2109	33.49	0.8379	0.0330	−2.6002	32.75
25,000	10,000	80	0.8350	0.0360	−2.3001	42.19	0.8400	0.0330	−2.6608	41.98
30,000	12,000	95	0.8360	0.0360	−2.3288	50.98	0.8410	0.0330	−2.6911	51.30

Note. Results for five training and scoring combinations are based on the average of 10 replications for C4.5, GAF, BGA, and BGAC.

sists 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 four are numerical. The proportion of the majority class (those who do not accept the promotion) is 83.50%. Using a similar rationale to that used for the adult data set, we set L = 0.5900, U = 0.9400, and F = 0.8350 for this data set.

In Tables 11, 12, and 13, we provide the analysis for this data set (these tables correspond to Tables 8, 9, and 10, respectively, for the adult data set). We see that the results from this data set point in exactly the same direction as the results from the other two data sets except that the improvement here is substantially more dramatic. Again, we conducted paired-difference t-tests to test for the improvements of BGAC over BGA; BGA over GAL, GAU, and GAF; and GAL, GAU, and GAF over C4.5. We found that all improvements were statistically significant with p-values of 0.0004 or less.

8. CONCLUSIONS

In this paper, we showed how to treat the classification accuracy of a decision tree as a random variable. We combined a tree's expected value and variance in a new probabilistic measure for assessing the performance of a tree. We developed a genetic algorithm for constructing a tree using our new measure (GA) and conducted computational experiments with three different variants (GAL, GAU, and GAF) and a standard decision tree package (C4.5) on three large data sets. In all three cases, GA outperformed C4.5 in terms of accuracy and z-score, and GAF-generated trees were, on average, more accurate than the trees generated by GAL and GAU.

We also investigated the effect of introducing diversity into the population used by our genetic algorithm. We allowed the genetic algorithm to simultaneously focus on two distinct probabilistic measures—one that is risk averse and one that is risk seeking—and tested two variants of our bivariate genetic algorithm (BGA and BGAC) on the three data sets. Our results showed that in terms of accuracy, depth, number of nodes, and z-score, BGAC outperformed BGA, which in turn outperformed GAF, which in turn outperformed C4.5.

Table 11. Computational results for C4.5, GAL, GAU, and GAF on the financial services data set.

	C4.5			GAL	GAU	GAF
Criterion	L	U	F	L	U	F
Accuracy	0.8820	0.8820	0.8820	0.8950	0.9039	0.9060
Standard error	0.0580	0.0580	0.0580	0.0460	0.0470	0.0450
Depth	8.0	8.0	8.0	6.4	6.5	6.6
Number of nodes	35.0	35.0	35.0	14.5	14.4	14.6
z-score	−5.0344	1.0000	−0.8103	−6.6306	0.7660	−1.5778
Time (minutes)	3.48	3.48	3.48	19.45	19.56	20.02

Note. L = 0.5900, U = 0.9400, and F = 0.8350.

Table 12. Computational results for BGA and BGAC on the financial services data set.

	BGA(L, U)			BGAC(L, U, F)
Criterion	L	U	F	F
Accuracy	0.9080	0.9119	0.9130	0.9200
Standard error	0.0390	0.0400	0.0390	0.0350
Depth	5.2	5.1	5.1	4.4
Number of nodes	9.7	9.8	10.0	7.8
z-score	−8.1539	0.6999	−2.0012	−2.4287
Time (minutes)	79.26	79.26	79.26	79.84

Note. L = 0.5900, U = 0.9400, and F = 0.8350.

Finally, we tested how four algorithms (C4.5, GAF, BGA, and BGAC) scaled up to handle large data sets. We found that GAF, BGA, and BGAC generated more accurate trees than C4.5 and their computation times increased linearly with the size of the training and scoring sets. Furthermore, our genetic algorithm, especially the BGAC variant, required only a small percent of the data to generate a high-quality decision tree.

The genetic algorithms appear to scale better than C4.5 because of the sampling approach we have applied within the genetic algorithm to generate the initial trees. Observe, for example, that when 99% of the transportation marketing data set is used, C4.5 is run on $310{,}000 + 124{,}000 = 434{,}000$ points. On the other hand, to generate the initial 200 trees within the genetic algorithms, C4.5 is run (200 times) on only 1,550 points. This is a key reason for the dramatic savings in running time for the large data sets. Within the genetic algorithm, a larger data set does not impact the running time of the genetic operators (i.e., performing crossover and mutation) except for scoring. However, scoring is a rather straightforward calculation. Consequently, the running times of the genetic algorithms increase at a much slower rate than the running times of C4.5.

The three data sets used for testing the algorithms were quite different in size, number of variables, and application domain. The adult data set was small, containing 48,842 data points and 14 variables. The transportation marketing data set was medium to large, containing about 440,000 data points and 11 variables. Finally, the financial services data set was very large, containing more than two million data points and 66 variables. The behavior of the genetic algorithms appears to be quite robust as the results for running time, accuracy, z-score, depth, and number of nodes is remarkably consistent across all three data sets. On all three data sets, the genetic algorithms consistently scale well, while BGAC outperforms BGA, which outperforms GAL, GAU, and GAF; each of which outperforms C4.5.

It is important to note that, in addition to an improved z-score (our objective), our genetic algorithms generate trees with improved accuracy, smaller depth, and fewer nodes. A smaller depth and fewer nodes indicate that the trees generated by our genetic algorithm are fairly small, containing fewer rules and simpler rules (i.e., there are fewer variables in a rule). Notice that we are obtaining these qualitative improvements in the simplicity of the rules generated by our decision tree without sacrificing either accuracy or z-score.

Our research demonstrates the benefits of using a genetic algorithm approach to build high-quality classification trees for very large data sets. Furthermore, it shows that increasing diversity in the population via fitness function perturbation yields a significant payoff with respect to building classification trees. We are hopeful that this research might

Table 13. Computational results for scaling experiments on the financial services data set.

Size			C4.5				GAF			
Training (N_t)	Scoring (N_s)	%	Accuracy	Standard Error	z-score	Time (min)	Accuracy	Standard Error	z-score	Time (min)
10,000	4,000	3	0.8830	0.0570	−0.8421	3.48	0.9060	0.0460	−1.5434	19.45
80,000	32,000	25	0.9080	0.0566	−1.2898	39.47	0.9201	0.0430	−1.9768	28.27
160,000	64,000	51	0.9140	0.0562	−1.4057	103.93	0.9230	0.0420	−2.0954	40.27
240,000	96,000	72	0.9160	0.0560	−1.4464	187.40	0.9240	0.0410	−2.1709	57.87
310,000	124,000	99	0.9170	0.0560	−1.4643	437.51	0.9249	0.0410	−2.1957	78.17

Size			BGA				BGAC			
Training (N_t)	Scoring (N_s)	%	Accuracy	Standard Error	z-score	Time (minutes)	Accuracy	Standard Error	z-score	Time (min)
10,000	4,000	3	0.9129	0.0400	−1.9501	79.40	0.9210	0.0360	−2.3890	79.72
80,000	32,000	25	0.9270	0.0380	−2.4213	87.42	0.9360	0.0350	−2.8860	89.04
160,000	64,000	51	0.9301	0.0370	−2.5677	101.85	0.9379	0.0340	−3.0298	103.49
240,000	96,000	72	0.9310	0.0370	−2.5952	120.77	0.9390	0.0340	−3.0598	120.02
310,000	124,000	99	0.9321	0.0360	−2.6950	146.17	0.9401	0.0330	−3.1822	147.62

Note. Results for five training and scoring combinations are based on the average of 10 replications for C4.5, GAF, BGA, and BGAC.

motivate vendors to develop genetic variants of their classification tree software for commercial application.

REFERENCES

Berry, M., G. Linoff. 1997. *Data Mining Techniques*. John Wiley and Sons, New York.

Blake, C., C. Merz. 1998. *UCI Repository of Machine Learning Databases*. Department of Information and Computer Science, University of California, Irvine, CA, available at www.ics.uci.edu/~mlearn/MLRepository.html.

Fu, Z. 2000. Using genetic algorithms to develop intelligent decision trees. Ph.D. dissertation, University of Maryland, College Park, MD.

——, B. Golden, S. Lele, S. Raghavan, E. Wasil. 2003. A genetic-algorithm-based approach for building accurate decision trees. *INFORMS J. Comput.* **15**(1) 3–22.

Gale, D., L. S. Shapley. 1962. College admissions and the stability of marriage. *Amer. Math. Monthly* **69** 9–15.

Jain, A., R. Duin, J. Mao. 2000. Statistical pattern recognition: A review. *IEEE Trans. Pattern Anal. Machine Intelligence* **22**(1) 4–37.

Lim, T.-S., W.-Y. Loh, Y.-S. Shih. 2000. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning* **40** 203–229.

Maulin, M. L. 1984. Maintaining diversity in genetic search. *Proc. Fourth National Conf. Artificial Intelligence*. AAAI Press, Menlo Park, CA, 247–250.

Michalewicz, Z. 1996. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, New York.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

Ripley, B. D. 1996. *Pattern Recognition and Neural Networks.* Cambridge University Press, Cambridge, U.K.

Whitley, D. 1989. The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. *Proc. Third Internat. Conf. Genetic Algorithms*. Morgan Kaufmann Publishers, San Mateo, CA, 116–121.