Available online at www.sciencedirect.com

ScienceDirect

Computational Statistics & Data Analysis 52 (2008) 1362 – 1372

COMPUTATIONAL STATISTICS & DATA ANALYSIS

www.elsevier.com/locate/csda

Classification tree analysis using TARGET

J. Brian Gray[a,*], Guangzhe Fan[b]

[a] Department of Information Systems, Statistics and Management Science, The University of Alabama, Tuscaloosa, AL 35487-0226, USA
[b] Department of Statistics and Actuarial Science, Center of Computational Mathematics for Industry and Commerce, University of Waterloo, Waterloo, Ont., Canada N2L 3G1

Received 17 May 2006; received in revised form 12 March 2007; accepted 12 March 2007
Available online 19 March 2007

Abstract

Tree models are valuable tools for predictive modeling and data mining. Traditional tree-growing methodologies such as CART are known to suffer from problems including greediness, instability, and bias in split rule selection. Alternative tree methods, including Bayesian CART (Chipman et al., 1998; Denison et al., 1998), random forests (Breiman, 2001a), bootstrap bumping (Tibshirani and Knight, 1999), QUEST (Loh and Shih, 1997), and CRUISE (Kim and Loh, 2001), have been proposed to resolve these issues from various aspects, but each has its own drawbacks.

Gray and Fan (2003) described a genetic algorithm approach to constructing decision trees called tree analysis with randomly generated and evolved trees (TARGET) that performs a better search of the tree model space and largely resolves the problems with current tree modeling techniques. Utilizing the Bayesian information criterion (*BIC*), Fan and Gray (2005) developed a version of TARGET for regression tree analysis. In this article, we consider the construction of classification trees using TARGET. We modify the *BIC* to handle a categorical response variable, but we also adjust its penalty component to better account for the model complexity of TARGET. We also incorporate the option of splitting rules based on linear combinations of two or three variables in TARGET, which greatly improves the prediction accuracy of TARGET trees. Comparisons of TARGET to existing methods, using simulated and real data sets, indicate that TARGET has advantages over these other approaches.
© 2007 Elsevier B.V. All rights reserved.

Keywords: CART; Data mining; Decision tree; Genetic algorithm; Predictive modeling; Random forests

1. Introduction

Classification and regression tree models are popular alternatives to the classical methods of discriminant analysis, logistic regression, and linear regression. The earliest decision tree models were automatic interaction detection (AID) proposed by Morgan and Sonquest (1963) and chi-square automatic interaction detection (CHAID) proposed by Kass (1980). Other methods followed, but the classification and regression trees (CART) methodology proposed by Breiman et al. (1984) is perhaps best known and most widely used.

CART and other recursive partitioning algorithms grow a binary decision tree in a sequential fashion by using splitting rules based on predictor variables to partition the data in such a way as to reduce the conditional variation in the response variable. In CART specifically, the trees are grown as large as possible to avoid early stopping, then pruned backward using a cost-complexity criterion to avoid overfitting of the training data and to obtain "right-sized"

* Corresponding author. Tel.: +1 205 348 8912; fax: +1 205 348 0560.
E-mail addresses: bgray@cba.ua.edu (J.B. Gray), gfan@uwaterloo.ca (G. Fan).

0167-9473/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.csda.2007.03.014

trees. CART uses cross-validation or a large independent test sample of data to select the best tree from the sequence of trees considered in the pruning process.

Researchers have identified several weaknesses in recursive partitioning tree construction methods. The main weakness is greediness in the tree-growing process. Similar to forward selection in variable selection for regression, local optimization at each step in the sequential node-splitting process does not guarantee global optimization of the overall tree structure. Breiman (2001b) has suggested that CART and similar tree techniques perform well with regard to interpretability, but not as well with regard to prediction. Breiman (1996a, 2001b) also discusses the problem of instability in CART solutions, where perturbing a small proportion of the training set or resampling the training set often results in a solution with a very different tree structure. Loh and Shih (1997) and Kim and Loh (2001, 2003) have also pointed out that, when choosing among splitting rules, CART is biased in favor of variables that have a large number of possible splits.

Several alternatives to CART have developed to address these problems. Bayesian CART (Chipman et al., 1998; Denison et al., 1998) and bootstrap bumping (Tibshirani and Knight, 1999) are stochastic search methods designed to avoid the greediness and instability problems with CART. QUEST (Loh and Shih, 1997) and CRUISE (Kim and Loh, 2001) are sequential tree-growing methods that are unbiased in choosing splitting rules and have the potential to improve on the greediness of CART. Breiman (1996b, 2001a), Freund (1995), and others proposed ensemble techniques, including bagging, random forests, and boosting, all of which combine information from a large number of trees grown from the data to make predictions. Empirical evidence suggests that ensembles of trees have better predictive capability than CART, but the simple interpretability of the single-tree model is lost.

Gray and Fan (2003) introduced tree analysis with randomly generated and evolved trees (TARGET), a non-greedy decision tree construction technique based on a genetic algorithm that performs a better search of the tree model space. Fan and Gray (2005) showed that regression trees constructed with the TARGET approach based on the Bayesian information criterion (*BIC*) have better performance than CART trees and only slightly worse performance than random forests, while maintaining a highly interpretable single-tree solution.

In this article, we describe the use of TARGET for constructing classification trees. In Section 2, we briefly describe the TARGET approach and how we modified the *BIC* used in Fan and Gray (2005) to handle a categorical response variable and to better account for the tree model complexity in the penalty term. To more fully explore the potential of TARGET, we also consider the use of split rules based on linear combinations of two or three variables. Examples comparing TARGET to alternative techniques are provided in Section 3. The empirical evidence suggests that TARGET has significant advantages over current classification tree methods.

2. The TARGET approach to constructing classification trees

The TARGET method for searching the tree model space is based on a genetic algorithm. Genetic algorithms are powerful yet simple search techniques originally developed by Holland (1975) for solving combinatorial optimization problems. Goldberg (1988) provides a detailed description of genetic algorithms. Genetic algorithms are beginning to make their way into statistical applications and methodology (see, e.g., Chatterjee et al., 1996; Wallet et al., 1996; Hamada et al., 2001; Baragona et al., 2001).

We assume that the data set is in standard form (as defined in Breiman et al., 1984) with a total of n observations. There is a categorical response variable Y, which is often binary in nature. There are p candidate predictor variables, $X_1, X_2, \ldots,$ and X_p, each of which can be categorical or quantitative in nature.

The TARGET methodology consists of the following steps: (1) initialization of a forest of randomly generated trees; (2) evaluation of the fitness of each tree in the forest; (3) evolution of the current forest to a new forest of trees using the genetic operations of crossover, mutation, cloning, and transplanting; and (4) repetition of steps (2) and (3) until improvement of the fitness values of the best trees in the forest has stabilized. We now describe these steps in greater detail.

2.1. *Initialization of the forest*

The initial forest is populated with 50 randomly grown classification trees by default. A randomly grown tree starts out as a single root node. At each stage of growing this tree, a terminal node is selected. With probability p_{split}, which is specified by the user, this node is split and then assigned a randomly generated split rule; otherwise, it remains a terminal node. A split rule consists of a split variable and a split set. The split rule is randomly generated for the node by randomly selecting a split variable from the list of potential predictor variables, then randomly choosing a split set based on the values of the split variable.

If the split variable is a quantitative or ordered variable, say X_s, with values $x_{s(1)} \leqslant x_{s(2)} \leqslant \cdots \leqslant x_{s(n)}$ in the data set, then the split set is determined by randomly choosing one of the values, say $x_{s(i)}$, and creating the split rule $X_s \leqslant x_{s(i)}$ for the node. For a given split variable X_s, there are n possible split sets (some of which are the same when there are ties among the values of X_s).

For quantitative variables, we also add the option of using linear combinations of two or three variables to construct a split rule. Following the notation in the previous paragraph, a two-variable combination is constructed as follows. Two variables X_{s1} and X_{s2} are randomly selected from the candidates with replacement. Two split values $x_{s1(i)}$ and $x_{s2(j)}$ are then randomly drawn for the selected variables, respectively. Two weights w_1 and w_2 are randomly and independently drawn from the interval $[-1, 1]$. The split rule is created as $w_1 X_{s1} + w_2 X_{s2} \leqslant w_1 x_{s1(i)} + w_2 x_{s2(j)}$. We could center and scale each quantitative variable to have mean 0 and standard deviation 1, but since the weights are randomly distributed on the interval $[-1, 1]$, there is no theoretical or practical reason to do so.

In the event that the split variable X_s is a nominal variable with values in the set of distinct values $C = \{c_1, c_2, \ldots, c_k\}$ within the data set, then the split set is determined by randomly dividing the values into two mutually exclusive groups, C_L and C_R, and creating the split rule $X_s \in C_L$ for the node. For a nominal split variable X_s with k categories, there are 2^{k-1} possible split sets.

The node-splitting probability p_{split} is used to control the average tree size in the initial forest. Large values of p_{split} result in larger randomly grown trees on average than those obtained from smaller values of p_{split}. The default value of p_{split} is 0.5 in TARGET.

2.2. *Evaluation of tree fitness values*

The fitness of a tree is measured by running the training observations in the data set through the tree and computing the overall tree fitness function (which is typically a function of the terminal node purities). Any of the node purity functions or tree criterion functions described in Breiman et al. (1984) can be utilized. For example, we might measure terminal node purity as the number of misclassified observations in a terminal node, and the tree fitness criterion could simply be the sum of the terminal node purities, i.e., the total number of observations misclassified by the tree. One major advantage of TARGET over CART and similar tree-growing methods is its flexibility in choosing a fitness criterion. Any reasonable, computable overall measure of tree performance can be used as a tree fitness function to be optimized. The overall criterion is not required to be a simple function of node purities.

Tree algorithms have a tendency to grow large trees and overfit the training data. In CART, cross-validation or a large test set is used in choosing the right-sized tree. In TARGET, we use the *BIC* to penalize model complexity and maintain reasonable tree sizes. The *BIC*, proposed by Schwarz (1978), is a likelihood-based statistical model selection criterion that penalizes for the size or complexity of the model.

In classification trees, we assume the conditional distribution of the response variable, given the values of the predictor variables, $X_1, X_2, \ldots$, and X_p, is a multinomial distribution. Suppose that there are K classes for the response variable y. Let T denote a tree with a set of terminal nodes $\{T\}$, and let $|T|$ denote the number of terminal nodes in tree T. The maximum likelihood estimate of the probability of each class k, $k = 1, 2, \ldots, K$, conditioned on being in terminal node $t \in \{T\}$ is

$$\hat{\pi}_{k|t} = \frac{n_{kt}}{n_t}, \tag{2.1}$$

where n_t is the number of observations in terminal node t and n_{kt} is the number of observations in terminal node t that belong to class k.

If we define $\ln L(T)$ as the log-likelihood of the training data for the tree model T and $\ln L(S)$ as the log-likelihood of the training data for a saturated model, then the deviance associated with terminal node t is

$$D(t) = 2 \ln L(S) - 2 \ln L(T) = 2 \sum_{k=1}^{K} n_{kt} \ln \left(\frac{n_t}{n_{kt}} \right). \tag{2.2}$$

The total deviance of tree model T is defined to be the sum of the deviances of the terminal nodes of T,

$$D(T) = \sum_{t \in \{T\}} D(t). \tag{2.3}$$

With regard to model complexity, we are estimating $K - 1$ probabilities for each terminal node, for a total of $(K-1)|T|$ parameters for the tree T. If the total number of observations in the training sample is n, then a *BIC* version of penalized deviance is

$$D_{\mathrm{BIC}}(T) = D(T) + (K-1)|T| \ln n. \tag{2.4}$$

See Fan and Gray (2005) for the analogous *BIC* version of penalized deviance in the regression tree context.

Chipman (2004) and Hastie et al. (2001) suggest that the effective number of parameters estimated may actually be much higher than $(K-1)|T|$ due to split rule selections made during the tree construction process. We decided to modify the penalty component in (2.4) to allow one additional degree of freedom for each split rule determined for the tree. The resulting modified *BIC* version of penalized deviance is given by

$$D_{\mathrm{BIC}}(T) = D(T) + [(K-1)|T| + (2|T|-1)] \ln n = D(T) + [(K+1)|T| - 1] \ln n. \tag{2.5}$$

Based on our empirical work to date, we have found that the modified version in (2.5) results in better predictive performance than our original version in (2.4).

2.3. *Evolution of the forest*

In the forest evolution step, trees for the new forest are created from trees in the current forest using the genetic operations of crossover, mutation, cloning, and transplanting. The new forest is the same size as the previous forest. The TARGET default is a forest of 50 trees.

In crossover, two trees in the current forest are selected at random, where the selection probability of a tree is proportional to its tree fitness value. The two trees then perform crossover to produce two new trees. In crossover, the two trees swap components. TARGET uses two types of crossover: (1) node swapping and (2) subtree swapping. In node swapping, we randomly identify a node in each of two randomly selected trees in the current forest, and then swap the splitting rules of these two nodes. All other features of the two trees remain the same. In subtree swapping, we randomly select one subtree (i.e., a randomly selected node and all of its descendant nodes) from each of two randomly selected trees in the current forest, and then swap the two subtrees. The TARGET defaults for crossover operations are: (1) to allow only the best single tree among the two parent and two child trees to be added to the new forest; and (2) create 30 trees for the new forest from crossover.

In mutation, one tree in the current forest is selected at random, where the selection probability of a tree is proportional to its tree fitness value. A single mutation is performed on the tree and the resulting tree is added to the new forest. We utilize four types of mutation: (1) split set mutation; (2) split rule mutation; (3) node swap mutation; and (4) subtree swap mutation. In split set mutation, a node in the tree is randomly selected and a new split set is randomly drawn for the split rule. In split rule mutation, a node in tree is randomly selected, a new split variable is randomly chosen, and a new split set for the new split variable is randomly created. For linear combination split rules, we also add the possibility of changing the value of one or more weights to mutate the split rule. Node swap and subtree swap mutations are analogous to their counterparts in crossover except that the changes are made within a single tree. In TARGET, the user specifies how many mutated trees are created at each generation, unlike traditional genetic algorithms, which perform mutation with a specified probability. The TARGET default for mutation is for 10 mutated trees to be added to the new forest.

In addition to the crossover and mutation operations, TARGET allows for two other features found in many genetic algorithms. Cloning, also known as elitism, ensures that good trees are not lost in the evolution process by making copies of the best trees to be included in the next generation. Transplanting, also known as immigration, allows for new randomly generated trees to be introduced into the next generation as a way of providing more variety or genetic material for the evolutionary process. In TARGET, the user specifies how many of the best trees are cloned for the next generation (TARGET default is five clones) and how many transplanted trees are added to the next generation (TARGET default is five transplants).

The new generation (forest) is composed of 50 trees obtained by applying these genetic operations. The training data are run through these trees and fitness values are computed. The evolutionary process is then repeated from generation to generation until there is little or no improvement in the best trees of the forest over several generations.

3. Examples and comparisons

In this section, we illustrate the use of TARGET with simulated and real data examples that have been considered elsewhere in the statistical literature. In these examples, we compare the performance of TARGET to RPART (Therneau and Atkinson, 1997) and random forests (Breiman, 2001a). RPART is representative of the greedy search algorithm CART, while random forests is a popular ensemble method. We also make some comparisons of TARGET to QUEST (Loh and Shih, 1997), CRUISE (Kim and Loh, 2001), Bayesian CART (Chipman et al., 1998; Denison et al., 1998), and bootstrap bumping (Tibshirani and Knight, 1999).

3.1. Comparisons with RPART and random forests

3.1.1. A high-order 4×4 interaction example

We consider a complicated high-order interaction problem by extending the 2×2 (Tibshirani and Knight, 1999) example to a 4×4 example. Here, the predictor variables X_1 and X_2 are independently and uniformly distributed on the interval [0, 4]. The variables X_3 through X_8 are noise variables that are uniformly distributed on the interval [0, 1]. The values of the binary response variable are shown in Fig. 1. The ideal tree model has 16 terminal nodes and uses X_1 and X_2 as predictor variables, ignoring X_3–X_8.

In the simulated data set, 1500 observations were generated from the model described above and randomly divided into a training set with 500 observations and a test set with 1000 observations. RPART, random forests, and TARGET were used to construct tree models on these data and their results were compared. For RPART, 10-fold cross-validation and 1-SE rule were used to select the tree. TARGET runs for 1000 generations in each trial. We also impose a minimum of five observations in each node for RPART and TARGET. For random forests, 100 trees are created for voting. To further test the performance of each method, γ% noise is added to the response variable Y, where $\gamma = 0$, 5, and 10, i.e., γ% of the observations are randomly selected and their values of Y are flipped from 0 to 1 or from 1 to 0. Results are summarized in Table 1 based on 50 realizations of the simulated data set for each noise level. From the results, we see that TARGET performs best in all situations. RPART, due to its greedy search nature, fails to identify the correct split rules. Random forests improves on the RPART performance by using many trees to vote, but its predictive performance is still not as good as that of TARGET's single-tree solution.

3.1.2. Another simulated data example

In this simulation, we use a tree structure that has linear combination split rules describing the simulated data. The predictor variables X_1–X_5 are randomly and uniformly distributed on interval [0, 1]. The true tree structure, with four

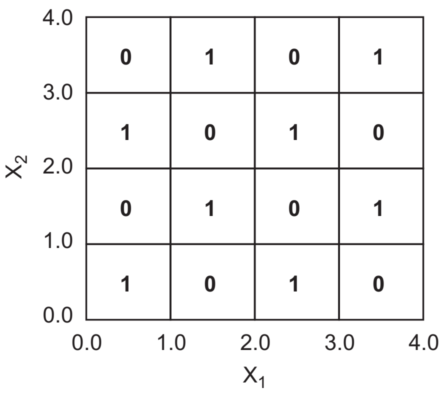

Fig. 1. Solution structure for the 4×4 interaction problem in Example 3.1.1.

Table 1
Summarized results for Example 3.1.1

Noise level	Method	Misclassification rate (std dev)	Tree size (std dev)
0%	TARGET	.08 (.10)	14.44 (3.12)
	RPART	.37 (.16)	16.16 (10.77)
	Random forests	.36 (.04)	—
5%	TARGET	.20 (.13)	12.08 (4.50)
	RPART	.45 (.10)	11.76 (9.30)
	Random forests	.41 (.03)	—
10%	TARGET	.35 (.12)	7.38 (4.38)
	RPART	.46 (.08)	9.88 (9.37)
	Random forests	.43 (.02)	—

For each method and noise level, the average test set misclassification rate and average tree size are reported for 50 realizations of the simulated data set. Standard deviations are shown in parentheses next to the averages.

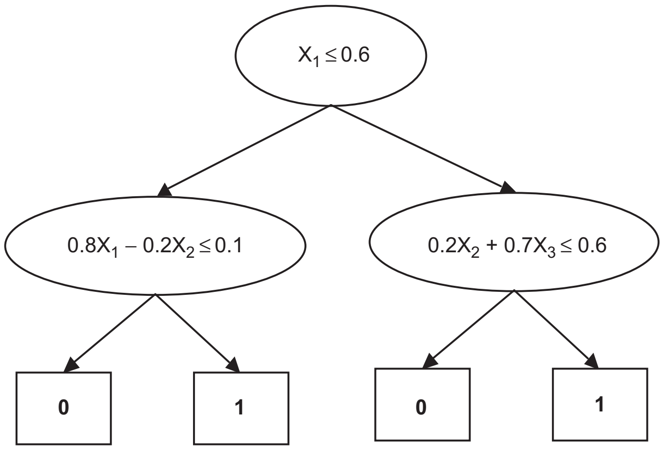

Fig. 2. Solution tree structure for Example 3.1.2.

terminal nodes as shown in Fig. 2, uses linear combinations of variables X_1, X_2, and X_3 to form split rules. Each terminal node is labeled with a Y value of 0 or 1. In the simulation, 1500 observations were generated from the model described above and randomly divided into a training set with 500 observations and a test set with 1000 observations. γ% noise is added to the Y variable, where $\gamma = 0$, 5, or 10, as in the previous example. Then, applying the same algorithm settings described in the example of Section 3.1.1, we use TARGET, TARGET-C2 (TARGET with the option of split rules based on linear combinations of two predictor variables), RPART, and random forests to fit models on the training data and report performance on the test set. The results from 50 simulation runs are summarized in Table 2. We see that TARGET-C2 works best under a variety of noise levels in the data.

3.1.3. Real data sets

We also considered three real data sets for comparison purposes. The first two data sets are from the UCI databases web site (ftp://ftp.ics.uci.edu/pub/machine-learning-databases/).

The Wisconsin breast cancer data (Mangasarian and Wolberg, 1990) has been used for illustration purposes in several tree-based methodology papers, including Chipman et al. (1998); Denison et al. (1998), and Zhang (1998). The data consist of 683 complete observations on nine predictor variables, having integer values ranging between 1 and 10, and a binary response variable (benign and malignant cases).

Table 2
Summarized results for Example 3.1.2

Noise level	Method	Misclassification rate (std dev)	Tree size (std dev)
0%	TARGET	.052 (.010)	9.78 (1.09)
	TARGET-C2	.034 (.014)	7.02 (1.39)
	RPART	.056 (.012)	8.82 (2.25)
	Random forests	.046 (.009)	—
5%	TARGET	.101 (.020)	8.02 (1.12)
	TARGET-C2	.078 (.019)	5.80 (1.16)
	RPART	.105 (.017)	7.59 (1.94)
	Random forests	.095 (.015)	—
10%	TARGET	.158 (.020)	6.98 (1.27)
	TARGET-C2	.128 (.018)	5.32 (1.00)
	RPART	.156 (.016)	6.54 (1.64)
	Random forests	.149 (.013)	—

For each method and noise level, the average test set misclassification rate and average tree size are reported for 50 realizations of the simulated data set. Standard deviations are shown in parentheses next to the averages.

Table 3
Summarized results for the three real data sets in Example 3.1.3

Data	Method	Misclassification rate (std dev)	Tree size (std dev)
Breast cancer	TARGET	.045 (.019)	5.80 (.45)
	TARGET-C2	.040 (.022)	4.58 (.62)
	TARGET-C3	.039 (.025)	4.27 (.66)
	RPART	.057 (.015)	3.80 (1.10)
	Random forests	.034 (.005)	—
Pima Indian	TARGET	.230 (.056)	5.64 (.64)
	TARGET-C2	.240 (.055)	5.49 (.75)
	TARGET-C3	.237 (.059)	5.15 (.89)
	RPART	.253 (.056)	4.31 (2.43)
	Random forests	.231 (.064)	—
Thyroid gland	TARGET	.064 (.058)	4.96 (.32)
	TARGET-C2	.038 (.041)	3.61 (.53)
	TARGET-C3	.044 (.043)	3.34 (.48)
	RPART	.098 (.045)	3.80 (.84)
	Random forests	.042 (.023)	—

For each method and data set, the average test set misclassification rate and average tree size are reported for 100 random training/test set partitions of the data. Standard deviations are shown in parentheses next to the averages.

The thyroid gland data set has 215 complete observations. The response variable has three classes: normal, hypo, and hyper. There are five predictor variables.

The third real data set is the Pima Indian data, which has 532 observations and can be found at http://www.stats.ox.ac.uk/pub/PRNN. A description of the data set can be found on the web site.

For comparison purposes, we adopt a popular strategy in the predictive modeling literature. Each data set is randomly partitioned into two parts, with 90% of the data used as a training set and 10% of the data used as a test set. This random partitioning is performed 100 times. The test set results are averaged over the 100 random partitions. The settings of each method are the same as those described in the example of Section 3.1.1. We allow TARGET, TARGET-C2, and TARGET-C3 (which allows for split rules based on linear combinations of three variables) to evolve for 1000 generations on each of the 100 training sets. Summarized results for the three data sets are shown in Table 3. From the table, we see that TARGET consistently outperforms RPART. The performance of TARGET is not quite as good as that of random forests (with the exception that the TARGET performs slightly better in the Pima Indian data), but its single-tree solutions are much easier to interpret than ensembles of trees. When linear combination split rules are added, we see that the predictive performance of TARGET is greatly improved and is close to or better than that of random forests, while still maintaining a single-tree structure with high interpretability. We note that in Pima Indian data, the single-variable split rules are sufficient for TARGET; the linear combination split rules do not provide further improvement. In practice, we recommend using 10-fold cross-validation to determine whether the option of linear combination split rules should be used.

3.2. Comparisons with QUEST and CRUISE

QUEST (Loh and Shih, 1997) and CRUISE (Kim and Loh, 2001, 2003) are methods that use unbiased statistical tests and linear discriminant analysis in finding split rules or in model fitting within the terminal nodes. Like the CART approach, trees are built in a sequential, stepwise manner by both QUEST and CRUISE, however, these methods have the potential for improving on CART's greediness in searching for split rules. They also have the potential of fitting a more suitable tree model or "treed" model based on the learning sample when the statistical assumptions are satisfied.

We use the two simulated data examples and the breast cancer data to compare QUEST and CRUISE to TARGET. For the simulated data example in Section 3.1.1, we use CRUISE with default settings specified by the program, but changed the minimum number of observations per node to 5. The experiment is repeated 50 times with 500 training observations and 1000 test observations. The results are summarized in the top portion of Table 4. The average test set performance is slightly better than for RPART and random forests, but not as good as that of TARGET. We observe that in roughly 40% of the trials, CRUISE stops at a root node tree; while in about 20% of the trials, the CRUISE performance is close to the expected solution of the simulated structure. We note that due to the XOR nature of the data, even with multi-way splits, the solution is still not easily identified by sequential tree-growing approaches.

For the simulated data example in Section 3.1.2, we use the Kim and Loh (2003) CRUISE approach. To implement this version of CRUISE, we chose the "univariate splits with bivariate node models" setting to grow the tree with other settings at their defaults. The results are summarized in the middle portion of Table 4. We see that CRUISE is not showing superiority in this example over TARGET or CART. One possible explanation might be the presence of the extraneous predictor variables and the difficulty in finding the correct first split rule.

We next use the breast cancer data from Section 3.1.3 to illustrate the performance of QUEST. The data are randomly split into a training set with 90% of the observations and a test set with the other 10% of the observations. We used QUEST's settings of "unbiased statistical tests" for variable selection and "discriminant analysis" for split point selection. Other settings are at their defaults. The experiment is conducted 100 times, and the average test set results are shown in the bottom portion of Table 4. We see that the QUEST result is superior to RPART and slightly better than that of TARGET. These results suggest that QUEST and CRUISE potentially improve on the

Table 4
Comparisons of TARGET to CRUISE and QUEST

Data	Method	Misclassification rate (std dev)	Tree size (std dev)
Example 3.1.1	CRUISE	.35 (.15)	15.62 (8.39)
	TARGET	.08 (.10)	14.44 (3.12)
Example 3.1.2	CRUISE	.068 (.017)	6.88 (2.02)
	TARGET	.052 (.010)	9.78 (1.09)
Breast cancer	QUEST	.042 (.023)	5.43 (1.91)
	TARGET	.045 (.019)	5.80 (.45)

For each method, the average test set misclassification rate and average tree size are reported. Standard deviations are shown in parentheses next to the averages.

CART methodology from a statistical point of view. However, to search for some specific tree structure, TARGET has the advantage of implementing a global search that does not rely heavily on the statistical assumptions about the data.

3.3. Comparisons with Bayesian CART and bootstrap bumping

Bayesian CART (Chipman et al., 1998; Denison et al., 1998) and bootstrap bumping (Tibshirani and Knight, 1999) are stochastic search methods for constructing a single-tree model. Bayesian CART uses Monte Carlo Markov chain (MCMC) as the search scheme with subjectively determined prior settings. According to the authors, the prior settings are essential for the success of the search. Bootstrap bumping uses CART's greedy tree construction while stochastically perturbing the training data by bootstrapping. To implement bumping, the recommended approach by the authors is to first run CART on the training data to determine the complexity parameter by cross-validation. Then this setting is used for growing bootstrap trees.

On the surface, TARGET and Bayesian CART appear to be fairly similar, but in reality, their stochastic searches over the space of tree models are quite different. Bayesian CART starts with a single tree and, through random changes made to that single tree, proceeds to sequentially visit trees local to that region of the tree space. Many restarts of this process are required in order to effectively search the entire tree space. TARGET starts with many randomly generated tree models scattered throughout the tree space and, through random genetic operations on that population of trees, performs parallel searches in many regions of the tree space. As the evolution process continues, the population of trees concentrates in more promising regions of the tree space. On that basis, we expect TARGET to perform a more efficient search of the tree space than Bayesian CART.

We first compare TARGET to Bayesian CART and bootstrap bumping on the basis of previously published results for which the other authors tuned their algorithms for performance. In a simulated tree structure proposed by Chipman et al. (1998), Bayesian CART required a median of 3000 trees to identify the correct root split. For the same problem, Tibshirani and Knight (1999) reported that the third quartile of the number of trees required by bootstrap bumping to identify the correct root split is around 600 trees for bootstrapping bumping. With the parallel search nature of TARGET, the probability of identifying the correct root split rule for this data set within the first generation of 50 random trees is $1-(\frac{15}{16})^{50}=96\%$. This gives TARGET a significant head-start advantage over Bayesian CART and bootstrap bumping in this example.

To further illustrate their relative performances in real world applications, we again use the breast cancer data of Section 3.1.3. We allowed Bayesian CART to run 3500 steps with 1000 prior draws following the default prior settings. We allowed TARGET to run for only 50 generations so that TARGET and Bayesian CART would visit roughly the same number of random trees. Bootstrap bumping builds trees using CART's greedy algorithm, which is not equivalent to visiting random trees. We allowed bumping to use 50 bootstrap trees. The data set is randomly split into a training set with 90% of the observations and a test set with the other 10% of the observations. We repeat the partitioning 50 times and the average test set performance is summarized in Table 5.

We observe that the three methods work similarly well in prediction for this data set. We also note that TARGET has a smaller average tree size, roughly one terminal node less than that of the other two methods. Smaller trees usually have better interpretability. We also notice that TARGET has less variability in its tree sizes than Bayesian CART, and much less than bootstrap bumping. This is partly due to the fact that the CART procedure used by bootstrap bumping has high instability to data perturbations.

Table 5
Comparisons of TARGET to Bayesian CART and bootstrap bumping

Data	Method	Misclassification rate (std dev)	Tree size (std dev)
Breast cancer	Bayesian CART	.045 (.024)	6.52 (.61)
	Bootstrap bumping	.046 (.022)	6.52 (1.71)
	TARGET	.045 (.019)	5.80 (.45)

For each method, the average test set misclassification rate and average tree size are reported. Standard deviations are shown in parentheses next to the averages.

4. Discussion

TARGET represents a major departure from current tree modeling techniques. A genetic algorithm is applied to search over the space of trees by initializing a forest of randomly generated trees, then evolving the forest through the genetic operations of crossover, mutation, cloning, and transplanting to improve the performance of the trees in the forest. Empirical evidence suggests that TARGET, with the modified *BIC*, can more effectively and thoroughly search the tree space and provide a single-tree solution with better predictive accuracy than greedy search algorithms, such as CART, and stochastic search algorithms, such as Bayesian CART. The single-tree solution that TARGET provides may not be as accurate as ensemble method such as random forests, but it is far more interpretable. With the option of split rules based on linear combinations of two or three variables, TARGET is better able to capture complicated structures in the data and have predictive accuracy closer to or better than that of ensemble methods such as random forests, while still keeping the interpretability of a single tree. We are currently investigating other forms of split rules based on multiple variables.

Another advantage of the TARGET approach is the flexibility of the choice of the node impurity measure and overall tree fitness function. Although the *BIC* setting performs well currently, other user-defined fitness functions can be adopted for specific use. For example, one could design a tree fitness function that would require highly accurate prediction for most nodes but allow for worse prediction in a few nodes (e.g., minimize the 95th percentile of the node misclassification rates). We are currently exploring a number of different node impurity measures and tree fitness criteria for use in TARGET.

It is interesting to compare the computational focus of CART and TARGET. In CART, most of the computational effort is spent in the tree-growing process looking for the optimal splitting rule at each node. In TARGET, however, tree growing is random and fast. The computational effort in TARGET is spent in the evaluation of trees resulting from the evolution process. We are currently working on approaches for saving time and storage in TARGET.

Our current implementation of TARGET in Java is slower than CART in running time, but the TARGET running times are reasonable for the problem sizes that we have considered. Our Java version of TARGET can take advantage of the parallel nature of TARGET by utilizing multiple processors, if they are available. We currently only re-evaluate the tree at the point where mutation or crossover takes place to save on computation. We are also looking at other ways of improving the running time of TARGET for larger problems.

References

Baragona, R., Battaglia, F., Calzini, C., 2001. Genetic algorithms for the identification of additive and innovation outliers in time series. Comput. Statist. Data Anal. 37, 1–12.

Breiman, L., 1996a. The heuristics of instability and stabilization in model selection. Ann. Statist. 24, 2350–2383.

Breiman, L., 1996b. Bagging predictors. Mach. Learn. 24, 123–140.

Breiman, L., 2001a. Random forests. Mach. Learn. 45, 5–32.

Breiman, L., 2001b. Statistical modeling: the two cultures. Statist. Sci. 16, 199–231.

Breiman, L., Friedman, J., Olshen, R., Stone, C., 1984. Classification and Regression Trees. Wadsworth, Belmont, CA.

Chatterjee, S., Laudatto, M., Lynch, L.A., 1996. Genetic algorithms and their statistical applications: an introduction. Comput. Statist. Data Anal. 22, 633–651.

Chipman, H., 2004. Personal communication.

Chipman, H.A., George, E.I., McCulloch, R.E., 1998. Bayesian CART model search (with discussions). J. Amer. Statist. Assoc. 93, 935–960.

Denison, D.G.T., Smith, A.F.M., Mallick, B.K., 1998. Comment. J. Amer. Statist. Assoc. 93, 954–957.

Fan, G., Gray, J.B., 2005. Regression tree analysis using TARGET. J. Computat. Graphical Statist. 14, 206–218.

Freund, Y., 1995. Boosting a weak learning algorithm by majority. Inform. and Comput. 121, 256–285.

Goldberg, D., 1988. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA.

Gray, J.B., Fan, G., 2003. TARGET: tree analysis with randomly generated and evolved trees. Technical Report, Applied Statistics Program, The University of Alabama.

Hamada, M., Martz, H.F., Reese, C.S., Wilson, A.G., 2001. Finding near-optimal Bayesian experimental designs via genetic algorithms. Amer. Statist. 55, 175–181.

Hastie, T., Tibshirani, R., Friedman, J., 2001. The Elements of Statistical Learning, Data Mining, Inference and Prediction. Springer, New York.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI.

Kass, G.V., 1980. An exploratory technique for investigating large quantities of categorical data. Appl. Statist. 29, 119–127.

Kim, H., Loh, W.-Y., 2001. Classification trees with unbiased multiway splits. J. Amer. Statist. Assoc. 96, 589–604.

Kim, H., Loh, W.-Y., 2003. Classification trees with bivariate linear discriminant node models. J. Comput. Graphical Statist. 12, 512–530.

Loh, W.-Y., Shih, Y.-S., 1997. Split selection methods for classification trees. Statist. Sin. 7, 815–840.

Mangasarian, O.L., Wolberg, W.H., 1990. Cancer diagnosis via linear programming. SIAM News 23, 1.
Morgan, J.N., Sonquest, J.A., 1963. Problems in the analysis of survey data and a proposal. J. Amer. Statist. Assoc. 58, 415–435.
Schwarz, G., 1978. Estimating the dimension of a model. Ann. Statist. 6, 461–464.
Therneau, T.M., Atkinson, E.J., 1997. An introduction to recursive partitioning using the RPART routines. Technical Report, Mayo Foundation.
Tibshirani, R., Knight, K., 1999. Model search and inference by bootstrap bumping. J. Comput. Graphical Statist. 8, 671–686.
Wallet, B.C., Marchette, D.J., Solka, J.L., Wegman, E.J., 1996. A genetic algorithm for best subset selection in linear regression. Proceedings of the 28th Symposium on the Interface of Computing Science and Statistics, vol. 28. pp. 545–550.
Zhang, H., 1998. Comment. J. Amer. Statist. Assoc. 93, 948–950.