Regression Tree Analysis Using TARGET

Guangzhe Fan and J. Brian Gray

Department of Information Systems, Statistics, and Management Science
The University of Alabama
Tuscaloosa, AL 35487-0226
Authors’ Footnote

Guangzhe Fan is Ph. D. candidate, Applied Statistics Program, The University of Alabama, Tuscaloosa, AL 35487-0226 (email: fan001@bama.ua.edu); and J. Brian Gray is Professor, Applied Statistics Program and Department of Information Systems, Statistics, and Management Science, The University of Alabama, Tuscaloosa, AL 35487-0226 (email: bgray@cba.ua.edu).
ABSTRACT

Most existing techniques for constructing regression trees are sequential in nature and locally optimal at each node split, but the final tree solution found may not be the best overall. In addition, small changes in the training data often lead to large changes in the final result due to the relative instability of these greedy tree-growing algorithms. Genetic algorithms are heuristic search techniques that start from a population of random solutions which is gradually evolved toward a globally optimum solution. Gray and Fan (2003) described the use of the TARGET (Tree Analysis with Randomly Generated and Evolved Trees) method to construct classification trees and compared its performance to CART and bagging. In this article, we show how the TARGET approach can also be used to construct regression trees for problems in which the response variable is quantitative. Simulated data and real world data are used to illustrate the process and compare its performance to CART. The empirical results indicate that TARGET regression trees are smaller than CART regression trees and have better test set performance.

KEYWORDS: CART, data mining, genetic algorithm, prediction
1. INTRODUCTION

TARGET (Tree Analysis with Randomly Generated and Evolved Trees) is a new tool in tree-structured predictive modeling introduced by Gray and Fan (2003). TARGET is a genetic search algorithm that begins with a randomly created population of tree models which is evolved toward an optimal solution. Gray and Fan (2003) focus on the use of TARGET to construct classification trees, i.e., when the response variable is categorical. This article addresses the use of TARGET for constructing regression trees, i.e., when the response variable is quantitative.

Recursive partitioning algorithms, such as CART (Breiman et al 1984), perform a greedy search for the best split at each stage when growing the tree. However, a search for the best split at each node level of the tree is not guaranteed to find the best overall tree. Cross-validation or a test sample is used to select the right-sized tree so that the model will perform as well on future data.

Recently, some authors have provided ways to improve tree-based models. Breiman (1996, 2001) proposed the methods of bagging (bootstrap aggregating) and random forests which yield complex, multiple-tree solutions. These methods improve prediction accuracy, but at the loss of single-tree interpretability. Tibshirani and Knight (1999) proposed a bootstrap “bumping” search that selects the best of a set of greedy trees grown on many bootstrap samples. Chipman et al (1998) provided a Bayesian CART model using a Monte Carlo Markov Chain stochastic search.

2. GENETIC ALGORITHMS FOR REGRESSION TREES

Genetic algorithms were first developed by Holland (1975). They are search algorithms based on the mechanics of natural selection and natural genetics. In each generation, a new
population of potential solutions (for example, trees) is created by combining and modifying members of the previous generation. While randomized, genetic algorithms are not simply a random walk; they efficiently exploit historical information to speculate on new potential solutions with expected improved performance (Goldberg 1989, chap. 1). Basically there are three aspects to using genetic algorithms: (1) evaluating a regression tree, (2) initializing a set of trees, and (3) evolving the trees using genetic operations.

2.1 Regression Tree Evaluation

Suppose we have \( p \) predictor variables \( X_1, X_2, \ldots, X_p \) and a response variable \( Y \). There are \( n \) observations in the learning sample, \( (x_i, y_i) \), for \( i = 1, 2, \ldots, n \), where \( x_i = (x_{i1}, x_{i2}, \ldots, x_{ip}) \). A regression tree with \( K \) leaves (terminal nodes) is constructed to partition the predictor variable space into \( K \) regions, \( T_k \), for \( k = 1, 2, \ldots, K \). The predictive model is given by

\[
f(x) = \bigoplus_{k=1}^{K} C_k \cdot I(x \in T_k),
\]

(2.1)

where \( C_k \) is a constant for each \( k \) and \( I(x \in T_k) \) is an indicator function. So equation (2.1) predicts \( Y \) as a constant value within each of the hyper-rectangular regions represented by the terminal nodes of the regression tree. Using a least squares criterion, we want to minimize the residual sum of squares

\[
SSE = \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2,
\]

(2.2)

where

\[
\hat{f}(x) = \bigoplus_{k=1}^{K} \hat{C}_k \cdot I(x \in T_k)
\]

(2.3)

and
\[ \hat{C}_k = \text{average}(y_i \mid x_i \in T_k), \quad k = 1, 2, \ldots, K. \]  

(2.4)

So we simply use the average of the \( Y \) values of the observations in the training sample that fall into a particular leaf node as the prediction of a future observation that falls into the node. Now we employ some notation that is very similar to that of regression to evaluate regression trees. We denote the total sum of squares of the observations as

\[ \text{SST} = \sum_{i=1}^{n} (y_i - \bar{y})^2. \]  

(2.5)

The \( R^2 \) value is computed as

\[ R^2 = 1 - \frac{\text{SSR}}{\text{SST}}. \]  

(2.6)

If we have a test sample that is independent of the training sample, we can also compute the test sample \( R^2 \) value, which provides a more “honest” estimate of the tree performance.

2.2 Initialization

For initialization, we randomly create \( n_f \) regression trees. At each node level, we randomly decide to split this node or to make it a terminal node. If the node is to be split, then a split rule, which includes a split variable and a split set, is randomly chosen from all candidate split rules and assigned to that node. The random tree-growing process eventually stops. The node splitting probability is used to control the expected tree size. After the tree is randomly created, the training data are run through the tree to determine their terminal nodes and fitted values.

Since the split rules are random, some nodes might be empty or have too few observations. These small or empty nodes are pruned from the tree before evaluating the trees.
To evaluate the trees, a fitness function must be specified. One of the major advantages of TARGET over existing tree-growing methods is that the choice of the fitness function is highly flexible. For example, we can use \( f(tree) = R^2 \) as a tree fitness function. To penalize larger trees that might be over-fitting, we could use \( f(tree) = R^2 - aK \) as a fitness function, where \( a \) is a complexity parameter specified by the user and \( K \) denotes the number of leaves in the tree. It is also easy to set the fitness function such that some other criterion, e.g., least sum of absolute residuals regression, is applied to the tree construction. After a fitness function is selected, the major task of the genetic algorithm is to optimize the tree fitness through genetic evolution.

2.3 Genetic Evolution

We call the randomly initialized trees the first generation. The following genetic operations are performed on randomly selected trees. Trees with larger fitness values are given higher probability of being selected for a genetic operation.

- **Elitism.** A number of trees with the best fitness values in the current generation are copied to the next generation.

- **Crossover.** Two trees are randomly selected and a node is randomly chosen on each tree. Then, either the two nodes are swapped or the two sub-trees are swapped to get two new trees that are sent to the next generation. In node-swap crossover, it is only the split rules that are actually swapped. We do not perform crossover if two root nodes or two leaves are selected.

- **Mutation.** Mutation can help the genetic search process avoid getting trapped at local optima. We randomly select a single tree and perform one of four types of mutation: (1)
randomly change a split set of the split rule of a randomly selected node while keeping the same split variable; (2) randomly change the split rule, including both the variable and its split set, of a randomly chosen node; (3) swap two randomly chosen nodes; and (4) swap two randomly chosen sub-trees within the tree. Probabilities corresponding to the four types of mutation can be specified by the user.

- **Transplant.** A number of new randomly generated trees are added to the next generation. This adds more “genetic material” to the process and provides more opportunities to find the global solution and to avoid local optima.

The new generation is composed of trees obtained by the above operations. The data is run through these trees and fitness values are computed. The evolutionary process is then repeated from generation to generation until an optimal solution is found or there is no improvement in the population of solutions.

### 3. EXAMPLES

In this section, we compare the performance of TARGET and CART on some real and simulated data examples.

#### 3.1 A Simulated Example

For this example, we simulate data as follows: $X_1$ and $X_2$ are independently and uniformly distributed on the interval $[0, 1]$. The response variable $Y$ is simulated as

$$Y = f(X_1, X_2) + \epsilon, \quad \epsilon \sim N(0,1),$$

where $f(X_1, X_2) = 3$, if both $X_1$ and $X_2$ are larger than 0.5 or smaller than 0.5; and $f(X_1, X_2) = -3$, otherwise. Some noise variables were also included in the data set:
\[ X_3 = X_1 + 2X_2 + 0.5d, \quad d \sim N(0,1), \]

\[ X_4 = X_1X_2, \]

\[ X_5 = X_1 \frac{d}{X_2}, \] and

\[ X_6 \sim Uniform(0,1). \]

An ideal tree model is expected to use only \( X_1 \) and \( X_2 \), have 4 terminal nodes, and have an \( R^2 \) value of approximately 90%.

We randomly generated 1000 observations from the model described above and split the data set into 5 subsets with 200 observations each, denoted by A, B, C, D and E. Each of the five subsets is then used as a training set of 200 observations to build a regression tree while the other 4 subsets are combined into a test set of 800 observations. We then used both CART 4.0 and TARGET to find regression trees from the five training data sets and compare their performances on the five test sets.

The results for CART are summarized in Tables 3.1, 3.2, and 3.3. Table 3.1 shows the results when only the two important predictor variables, \( X_1 \) and \( X_2 \), are considered by CART. The five CART trees found from the five training sets are much larger than the true four-node solution. Although the training set \( R^2 \) values are close to the true value of 90%, their test set \( R^2 \) values are off by as much as 10-20% from the training set performance for three of the five trees. The CART results in Table 3.2 are for the situation in which \( X_1, X_2, X_3 \), and \( X_6 \) are considered as candidate predictors. The CART trees are even larger in this case and the test performance is much worse with three trees having \( R^2 \) values less than 60%. Finally, Table 3.3 summarizes the CART results when all six predictor variables are candidates for selection. Once again, the CART trees are larger than expected and \( R^2 \) values for the test sets are well below the true...
value. All three tables also show that CART has difficulty finding the correct first split, \( X_1 \leq 0.5 \) (or \( X_2 \leq 0.5 \)), especially in the presence of noise variables or highly correlated variables.

Table 3.4 summarizes the TARGET results when all six predictor variables are candidates for selection. The regression trees found by TARGET are close to the true model used to generate the data. (The small deviations of the fitted models from the true model are due to the discreteness of the generated training data over the \((X_1, X_2)\) unit square.) The trees are small and easy to interpret. Figure 3.1 shows the evolution process when data set A was used as the training set and data sets B, C, D, and E were combined into a test set. We see that both training and testing \( R^2 \) values are gradually increasing and tracking each other fairly closely. This suggests less likelihood of overfitting in the TARGET solutions.

3.2 MPG data

The data used in this example were taken from the University of California-Irvine Machine Learning Repository (ftp://ftp.ics.uci.edu/pub/machine-learning-databases/). The total number of observations is 398, with 6 observations having missing values. We use the 392 complete observations in our example. The response variable is MPG (miles per gallon) and the
predictor variables are Cylinders, Displacement, Horsepower, Weight, Acceleration, Year, and Origin. The data is divided randomly into two subsets, 262 observations as a training set and 130 observations as a test set. The results of CART and TARGET analyses are shown in Table 3.5.

The results show that TARGET found trees with good performance and small size. CART makes a little improvement on its performance with a much larger tree of 21 leaves. But such a large tree is more difficult to interpret.

For trees found by TARGET, the training and test set performances, as measured by $R^2$, are very close, suggesting that there is little evidence of overfitting. The reason is that instead of splitting the data and growing the tree, TARGET searches the entire tree space and locates tree structures that model the data well. Figures 3.2 and 3.3 give a clear picture of how the process evolves. The result is based on the best trees found at each generation.

We see from Figure 3.2 and Figure 3.3 that the best trees gradually increase in size with training and testing fitness values converging to each other.

3.3 Boston Housing Data

The data for this well known example are taken from the University of California-Irvine Machine Learning Repository (ftp://ftp.ics.uci.edu/pub/machine-learning-databases/). We try to
use the 13 predictor variables to predict the response variable, the median value (MV, in thousands of dollars) of homes in 506 census tracts in the Boston area. The 506 observations were randomly divided into two data sets, a training set of 338 observations and a test set of 168 observations. Once again, both CART and TARGET were used to grow trees. We repeated the experiment a second time by creating another random partition of the data into 338 training observations and 168 test observations. Results from both experiments are reported in Table 3.6.

- Insert Table 3.6 here -

We can see from the result again that TARGET finds trees with similar and strong predictive performance on the training and testing sets. The CART solutions have much wider gaps between the training and test performance. So the trees found by TARGET are perhaps more “honest” than the CART solutions. We also notice that CART found a large tree with 132 leaves that is certainly very difficult to interpret and is very likely to be overfitting the data.

- Insert Figures 3.4, 3.5, and 3.6 here -

4. CONCLUSION

TARGET, based on a genetic algorithm, is a competitive alternative to existing greedy tree-growing methods such as CART. The empirical evidence presented in this article suggests that TARGET regression trees are smaller than CART trees and have better test set performance. In addition, the TARGET results were seen to be more stable than the CART results when the input data were randomly partitioned into training and test sets.
REFERENCES


TABLES

Table 3.1. CART Results for Example 3.1: $X_1$ and $X_2$ Used as Predictor Variables.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Training R-square</th>
<th>Testing R-square</th>
<th>Number of Leaves</th>
<th>First Split</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>90.60%</td>
<td>80.60%</td>
<td>7</td>
<td>$X_1 \lessdot .04$</td>
</tr>
<tr>
<td>B</td>
<td>89.60%</td>
<td>69.00%</td>
<td>13</td>
<td>$X_1 \lessdot .98$</td>
</tr>
<tr>
<td>C</td>
<td>90.50%</td>
<td>80.70%</td>
<td>10</td>
<td>$X_1 \lessdot .90$</td>
</tr>
<tr>
<td>D</td>
<td>90.90%</td>
<td>84.50%</td>
<td>7</td>
<td>$X_2 \lessdot .78$</td>
</tr>
<tr>
<td>E</td>
<td>88.90%</td>
<td>87.20%</td>
<td>6</td>
<td>$X_2 \lessdot .46$</td>
</tr>
</tbody>
</table>

Table 3.2. CART Results for Example 3.1: $X_1$, $X_2$, $X_3$, and $X_6$ Used as Predictor Variables.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Training R-square</th>
<th>Testing R-square</th>
<th>Number of Leaves</th>
<th>First Split</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>91.50%</td>
<td>44.00%</td>
<td>22</td>
<td>$X_3 \lessdot .20$</td>
</tr>
<tr>
<td>B</td>
<td>44.80%</td>
<td>36.10%</td>
<td>9</td>
<td>$X_3 \lessdot 1.36$</td>
</tr>
<tr>
<td>C</td>
<td>82.00%</td>
<td>57.70%</td>
<td>13</td>
<td>$X_3 \lessdot 1.65$</td>
</tr>
<tr>
<td>D</td>
<td>90.90%</td>
<td>84.50%</td>
<td>7</td>
<td>$X_2 \lessdot .78$</td>
</tr>
<tr>
<td>E</td>
<td>90.60%</td>
<td>81.30%</td>
<td>13</td>
<td>$X_2 \lessdot .46$</td>
</tr>
</tbody>
</table>
Table 3.3. CART Results for Example 3.1: $X_1$, $X_2$, $X_3$, $X_4$, $X_5$, and $X_6$ Used as Predictor Variables.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Training R-square</th>
<th>Testing R-square</th>
<th>Number of Leaves</th>
<th>First Split</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>86.20%</td>
<td>69.50%</td>
<td>10</td>
<td>$X_5 \leq 0.43$</td>
</tr>
<tr>
<td>B</td>
<td>87.20%</td>
<td>85.50%</td>
<td>8</td>
<td>$X_4 \leq 0.47$</td>
</tr>
<tr>
<td>C</td>
<td>86.20%</td>
<td>51.80%</td>
<td>13</td>
<td>$X_4 \leq 0.41$</td>
</tr>
<tr>
<td>D</td>
<td>90.90%</td>
<td>64.70%</td>
<td>9</td>
<td>$X_4 \leq 0.34$</td>
</tr>
<tr>
<td>E</td>
<td>89.40%</td>
<td>64.60%</td>
<td>13</td>
<td>$X_5 \leq 0.41$</td>
</tr>
</tbody>
</table>

Table 3.4. TARGET Results for Example 3.1: $X_1$, $X_2$, $X_3$, $X_4$, $X_5$, and $X_6$ Used as Predictor Variables.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Training R-square</th>
<th>Testing R-square</th>
<th>Number of Leaves</th>
<th>First Split</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>88.78%</td>
<td>86.43%</td>
<td>4</td>
<td>$X_2 \leq 0.50$</td>
</tr>
<tr>
<td>B</td>
<td>88.00%</td>
<td>86.74%</td>
<td>4</td>
<td>$X_2 \leq 0.50$</td>
</tr>
<tr>
<td>C</td>
<td>88.59%</td>
<td>84.20%</td>
<td>4</td>
<td>$X_1 \leq 0.50$</td>
</tr>
<tr>
<td>D</td>
<td>90.41%</td>
<td>84.44%</td>
<td>4</td>
<td>$X_1 \leq 0.49$</td>
</tr>
<tr>
<td>E</td>
<td>87.33%</td>
<td>86.72%</td>
<td>4</td>
<td>$X_1 \leq 0.49$</td>
</tr>
</tbody>
</table>
Table 3.5. Comparison of CART and TARGET Results for MPG Data.

<table>
<thead>
<tr>
<th>Number of Leaves</th>
<th>Training R-square</th>
<th>Testing R-square</th>
</tr>
</thead>
<tbody>
<tr>
<td>CART</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21*</td>
<td>93.5%</td>
<td>85.1%</td>
</tr>
<tr>
<td>8</td>
<td>84.6%</td>
<td>81.0%</td>
</tr>
<tr>
<td>TARGET</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>83.4%</td>
<td>83.2%</td>
</tr>
<tr>
<td>7</td>
<td>81.2%</td>
<td>81.7%</td>
</tr>
</tbody>
</table>

*CART output indicates this tree to be optimal based on test set performance.

Table 3.6. Comparison of CART and TARGET Results for Boston Housing Data.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Method</th>
<th>Number of Leaves</th>
<th>Training R-square</th>
<th>Testing R-square</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CART</td>
<td>8*</td>
<td>85.4%</td>
<td>70.9%</td>
</tr>
<tr>
<td></td>
<td>TARGET</td>
<td>8</td>
<td>79.1%</td>
<td>75.2%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>132*</td>
<td>99.7%</td>
<td>83.5%</td>
</tr>
<tr>
<td>2</td>
<td>CART</td>
<td>8</td>
<td>86.7%</td>
<td>75.7%</td>
</tr>
<tr>
<td></td>
<td>TARGET</td>
<td>7</td>
<td>84.2%</td>
<td>73.8%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>79.5%</td>
<td>78.1%</td>
</tr>
<tr>
<td></td>
<td>TARGET</td>
<td>7</td>
<td>78.7%</td>
<td>77.4%</td>
</tr>
</tbody>
</table>

*CART output indicates this tree to be optimal based on test set performance.
Figure 3.1. Training and Testing R-square Values of the Best Tree at Each Generation During Evolution of TARGET on Training Data Set A.

Figure 3.2. Training and Testing R-square Values of the Best Tree at Each Generation in the Evolution on the MPG Data.
Figure 3.3. Number of Leaves of the Best Trees at Each Generation in the Evolution.
Figure 3.4. The 8-Leaves Tree Found by TARGET in Experiment 2 for the Boston Housing Data. The average (and standard deviation) of the response MV are shown in each leaf.
Figure 3.5. Training and Testing R-square Values of the Best Tree at Each Generation in the Evolution of TARGET for the Boston Housing Data in Experiment 1.

Figure 3.6. Training and Testing R-square Values of the Best Tree at Each Generation in the Evolution of TARGET for the Boston Housing Data for Experiment 2.