Optimal Decision Trees

Kristin P. Bennett
Rensselaer Polytechnic Institute
165 PUBLICATIONS **7,253** CITATIONS

Optimal Decision Trees

Kristin P. Bennett*
Jennifer A. Blue
Department of Mathematical Sciences
Rensselaer Polytechnic Institute
Troy, NY 12180

R.P.I. Math Report No. 214

Abstract

We propose an Extreme Point Tabu Search (EPTS) algorithm that constructs globally optimal decision trees for classification problems. Typically, decision tree algorithms are greedy. They optimize the misclassification error of each decision sequentially. Our non-greedy approach minimizes the misclassification error of all the decisions in the tree concurrently. Using Global Tree Optimization (GTO), we can optimize existing decision trees. This capability can be used in classification and data mining applications to avoid overfitting, transfer knowledge, incorporate domain knowledge, and maintain existing decision trees. Our method works by fixing the structure of the decision tree and then representing it as a set of disjunctive linear inequalities. An optimization problem is constructed that minimizes the errors within the disjunctive linear inequalities. To reduce the misclassification error, a nonlinear error function is minimized over a polyhedral region. We show that it is sufficient to restrict our search to the extreme points of the polyhedral region. A new EPTS algorithm is used to search the extreme points of the polyhedral region for an optimal solution. Promising computational results are given for both randomly generated and real-world problems.

Key Words: decision trees, tabu search, classification, machine learning, global optimization.

*Knowledge Discovery and Data Mining Group, Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180. Email bennek@rpi.edu, bluej@rpi.edu. This material is based on research supported by National Science Foundation Grant 949427.

1 Introduction

Decision trees have proven to be a very effective technique for classification problems. A training set, consisting of samples of points with n attributes from each class, is given. Then a decision tree is constructed to discriminate between these sets. The new decision tree is used to classify future points. The tree can be interpreted as rules for membership in the classes. Since decision trees have a readily interpretable logical structure, they provide insight into the characteristics of the classes.

We propose a non-greedy non-parametric approach to constructing multivariate decision trees that is fundamentally different than greedy approaches. Popular decision tree algorithms are greedy. Greedy univariate algorithms such as CART [9] and C4.5 [30] construct a decision based on one attribute at a time. Greedy multivariate decision tree algorithms such as those in [2, 10, 24] construct decisions one at a time using linear combinations of all the attributes. In each case, an optimization problem is solved at each decision node to determine the locally best decision. The decision divides the attribute space into two or more regions. Decisions are constructed recursively for each of the regions. This process is repeated until the points in each region (a leaf of the tree) are all, or almost all, of the same class. Using this approach, it is possible to construct a tree to discriminate between any two disjoint sets. However, the resulting tree may be overparameterized and thus may not reflect the underlying characteristics of the data set. When the overfitting occurs, the tree may not classify future points well. To avoid this problem, heuristics are applied to prune decisions from the tree [29, 22].

In this paper, optimization techniques are used to minimize the error of the entire decision tree. Our global approach is analogous to the widely used back propagation algorithm for constructing neural networks [31]. For a neural network, one specifies an initial structure, the number of units, and their interconnections. An error function which measures the error of the neural network is then constructed. The decisions in a multivariate decision tree are the same as linear threshold units in a neural network. In global tree optimization (GTO) an initial structure, the number of decision nodes, and the class of the leaf nodes are specified. We propose several possible error functions that measure the error of the entire decision tree. One of two different optimization methods, Frank-Wolfe and Extreme Point Tabu Search, can be used to minimize the error of the tree.

GTO combines the benefits of decision trees and neural network methods. The great strengths and challenges of decision trees are that they are logically interpretable but constructing them is combinatorially difficult. Typically, greedy methods are constructed one decision at a time starting at the root. Locally good but globally poor choices of the decisions at each node, however, can result in excessively large trees that do not reflect the underlying structure of the data. Pruning may not be sufficient to compensate for overfitting. This problem is readily shown in multivariate decision trees. The pruning process frequently produces a tree consisting of a single decision [2, 5, 33]. Univariate algorithms appear less susceptible to this problem. Murthy and Salzburg found that greedy heuristics worked well and lookahead algorithms offered little improvement [25, 26]. We believe that this is because univariate decision trees have only one degree of freedom at each decision. The problem with univariate trees, however, is that many decisions are required to represent

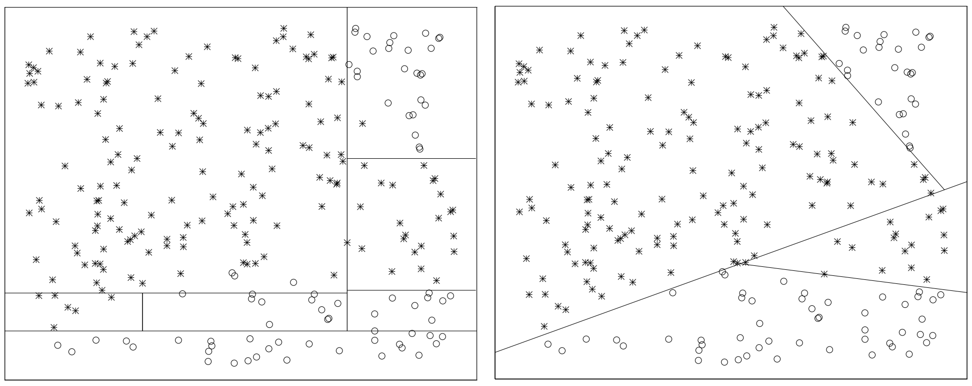


Figure 1: Performance of C4.5 and GTO on a sample problem.

simple linear relations. Multivariate decisions are much more powerful. But greedy methods are more easily led astray. In GTO, we search for the best multivariate tree with a given structure. Fixing the structure prevents the method from overfitting the data. An example of this can be seen in Figure 1. The algorithm C4.5 [30] and the GTO algorithm were both applied to this sample data set of 250 points in two dimensions. The C4.5 algorithm used six univariate decisions and still was unable to correctly classify all of the points. GTO applied to this data set with the initial tree structure of Figure 2 correctly classified all the points.

Another benefit of GTO is that existing trees may be optimized. Typically, greedy methods must reconstruct a tree from scratch each time the algorithm is run. The ability to optimize an existing tree can be very useful in classification and data mining applications. Domain knowledge can be incorporated into the initial tree and the tree then can be optimized. Knowledge transfer can be achieved by starting with a tree from a related problem or by updating an existing tree when new data becomes available. GTO can be used to prune and restructure a tree initially constructed via a greedy method. The error function used within GTO can be customized to meet a client's particular needs. This flexibility of our method is very promising. We are just beginning to explore these possibilities.

Since GTO is non-parametric, it is fundamentally different from the few prior nongreedy decision tree algorithms. Unlike the Bayesian or parametric approaches [32, 11, 18, 19], the strict logical rules of the tree are maintained. GTO is based on the idea of formulating a decision tree as a set of disjunctive linear inequalities. In Section 2, we show that the underlying problem of finding a decision tree with fixed structure that completely classifies two sets is equivalent to finding a solution of a set of disjunctive linear inequalities. In Section 3, we propose objective functions that can be used to minimize the error within the disjunctive inequalities. The problem thus becomes a nonlinear nonconvex optimization problem over a polyhedral region. In Section 4, we show that there exist extreme point solutions for the proposed optimization problems which are as good as or better than the optimal solutions to the problems in terms of the number of points misclassified.

In Sections 5 and 6 we propose two optimization algorithms for minimizing the error in the disjunctive inequalities by traversing the extreme point solutions. The Frank-Wolfe algorithm

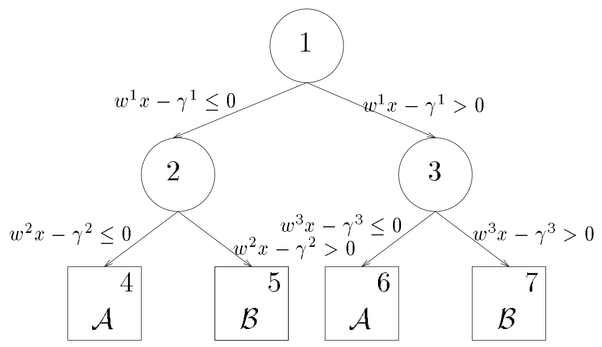


Figure 2: A multivariate decision tree with three decisions

(FW) is a descent technique that stops at the first local minimum it encounters. Extreme point tabu search (EPTS) is a global optimization technique. While it is not guaranteed to find the global minimum, it is much more robust than the Frank-Wolfe algorithm. In addition, the Frank-Wolfe algorithm is limited to differentiable objective functions. For EPTS, the objective function need not be differentiable or even continuous. Very strong computational results for both of these algorithms are given in Section 7.

We will use the following notation: For a vector x in the n-dimensional real space R^c, $(x)_i$ denotes the i^{th} component of x, and x_+ will denote the vector in R^c with components $(x_+)_i := \max\{x_i, 0\},\ i = 1, \ldots, c$. The dot product of two vectors x and w will be indicated by xw. The outer product is never used.

2 Decision Trees as Disjunctive Inequalities

Our goal is to formulate an optimization problem that can be used to construct a decision tree with given structure that recognizes points from two or more classes. The key idea is that a given multivariate decision tree can be represented as a set of disjunctive linear inequalities [7, 3]. If we can find a solution to the set of disjunctive linear inequalities, we have found a tree that correctly classifies all the points. Note that the structure of the tree must be fixed, but the actual decisions to be used are not.

The correspondence of the decision tree to the disjunctive inequalities is conceptually simple but notationly intricate. Consider the decision tree with three decisions shown in Figure 3. Each decision in the tree corresponds to a linear inequality. As a point traverses a path from the root of the tree to a classification leaf, each decision corresponds to an inequality that must be satisfied. Several leaves may belong to a given class. If a point satisfies any one of the sets of inequalities corresponding to a leaf of the appropriate class, then it is correctly classified. Thus we would like a set of disjunctive inequalities to be satisfied.

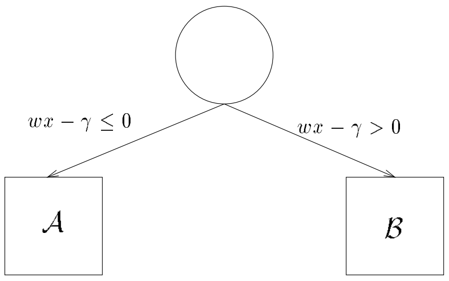


Figure 3: A multivariate decision tree with one decision

2.1 Sample Problems

In this paper, we restrict our discussion to the two-class discrimination problem using a binary multivariate decision tree. These results can be generalized to problems with three or more classes. Let there be two sets $\mathcal{A}$ and $\mathcal{B}$ containing m_A and m_B points, respectively. Each point has n real-valued attributes. If the attributes are not real-valued, the techniques used in neural networks for mapping symbolic attributes into real attributes can be used. Each decision consists of a linear combination of attributes. For the simple case of one decision as seen in Figure 3, let A_i be the i^{th} point in $\mathcal{A}$, w be the weights of the decision, and γ be the threshold. If $A_i w - \gamma < 0$ then the point follows the left branch of the tree. If $A_i w - \gamma > 0$ then the point follows the right branch of the tree. If $A_i w - \gamma = 0$ we use the convention that the point follows the left branch. We consider this situation undesirable and count it as an error when training. We will first investigate the simple cases of a decision trees consisting of one decision and three decisions. We will then give the formulation for general multivariate binary decision trees.

2.1.1 Decision Tree With One Decision

Consider the simplest case, a two-class decision tree consisting of a single decision in which the left leaf is labeled $\mathcal{A}$ and the right leaf is labeled $\mathcal{B}$. This corresponds to a linear discriminant such as the one shown in Figure 3. Let w be the weights of the decision and γ be the threshold. The decision accurately classifies all the points in $\mathcal{A}$ if there exist w and γ such that $A_i w - \gamma < 0$ for all points $i = 1, \ldots, m_A$. The decision accurately classifies all the points in $\mathcal{B}$ if $B_j w - \gamma > 0$ for all points $j = 1, \ldots, m_B$. We now remove the strict inequalities by adding a constant term. The inequalities become

$$A_i w - \gamma \ \leq \ -1 \quad i = 1, \ldots, m_A \quad and \quad B_j w - \gamma \ \geq \ 1 \quad j = 1, \ldots, m_B \tag{1}$$

We can easily determine in polynomial time if such a w and γ exist using a single linear program [6]. Notice that since there is no scaling on w and γ the constant (in this case 1) may be any positive number.

2.1.2 Decision Tree With Three Decisions

Now consider the problem of a tree with multiple decisions such as the three-decision tree in Figure 2. In order for the points in $\mathcal{A}$ to be correctly classified at leaf nodes 4 or 6, the following inequalities must be satisfied:

$$\begin{array}{c} \{A_i w^1 - \gamma^1 \ \leq \ -1 \ \ \textit{and} \ \ A_i w^2 - \gamma^2 \ \leq \ -1\} \\ \textit{or} \\ \{A_i w^1 - \gamma^1 \ \geq \ 1 \ \ \textit{and} \ \ A_i w^3 - \gamma^3 \ \leq \ -1\} \\ i = 1, \ldots, m_A \end{array} \tag{2}$$

In order for the points in $\mathcal{B}$ to be correctly classified at leaf nodes 5 or 7, the following inequalities must be satisfied:

$$\begin{array}{c} \{B_j w^1 - \gamma^1 \ \leq \ -1 \ \ \textit{and} \ \ B_j w^2 - \gamma^2 \ \geq \ 1\} \\ \textit{or} \\ \{B_j w^1 - \gamma^1 \ \geq \ 1 \ \ \textit{and} \ \ B_j w^3 - \gamma^3 \ \geq \ 1\} \\ j = 1, \ldots, m_B \end{array} \tag{3}$$

Thus if w and γ exist that satisfy disjunctive inequalities (2) and (3), the tree will correctly classify all the points.

2.2 General Case

This same approach can be used for any binary multivariate decision tree with fixed structure. The trees may be of arbitrary depth and need not be symmetric. First the structure of the tree must be specified. Let the tree have D decision nodes and $D+1$ leaf nodes. Assume the decision nodes are numbered 1 through D, and the leaf nodes are numbered $D+1$ through $2D+1$. Let θ_A be the set of leaf nodes classified as $\mathcal{A}$ and let θ_B be the set of leaf nodes classified as $\mathcal{B}$. Consider the path traversed from the root to a leaf node. For each leaf node k, define G^k as the index set of the decisions in which the right or "greater than" branch is traversed to reach leaf k. For each leaf node k, define L^k as the index set of the decisions in which the left or "less than" branch is traversed to reach leaf k. For example, the three-decision tree of Figure 2 has the following values:

$$\begin{array}{c} \theta_A \ = \ \{4,6\} \qquad \theta_B \ = \ \{5,7\} \\ L^4 \ = \ \{1,2\} \quad L^5 \ = \ \{1\} \quad L^6 \ = \ \{3\} \quad L^7 \ = \ \{\emptyset\} \\ G^4 \ = \ \{\emptyset\} \quad G^5 \ = \ \{2\} \quad G^6 \ = \ \{1\} \quad G^7 \ = \ \{1,3\} \end{array} \tag{4}$$

A point is correctly classified in $\mathcal{A}$ if all the inequalities from the root to a leaf node are satisfied for some leaf node of class $\mathcal{A}$. Similarly, a point is correctly classified in $\mathcal{B}$ if all the inequalities from the root to a leaf node of class $\mathcal{B}$ are satisfied. We define the correct classification of all the points as follows.

Definition 2.1 (Correct Classifications of Points by a Given Tree) *The set of disjunctive inequalities that must be satisfied in order to correctly classify each of the points* *in $\mathcal{A}$ for a tree with fixed structure is given by:*

$$\bigvee_{k\in\theta_A}\left\{\left\{\bigwedge_{d\in G^t}(A_i w^d-\gamma^d\geq 1)\right\}\bigwedge\left\{\bigwedge_{d\in L^t}(A_i w^d-\gamma^d\leq -1)\right\}\right\}\qquad i=1,\ldots,m_A \tag{5}$$

The set of disjunctive inequalities that must be satisfied in order to correctly classify each of the points in $\mathcal{B}$ for a tree with fixed structure is given by:

$$\bigvee_{k\in\theta_B}\left\{\left\{\bigwedge_{d\in G^t}(B_j w^d-\gamma^d\geq 1)\right\}\bigwedge\left\{\bigwedge_{d\in L^t}(B_j w^d-\gamma^d\leq -1)\right\}\right\}\qquad j=1,\ldots,m_B \tag{6}$$

Thus, if there exist w and γ such that the disjunctive inequalities of (5) and (6) are satisfied, then all the points are classified correctly by the tree.

For the general case, once the initial tree structure is chosen, the complete set of disjunctive inequalities can be explicitly defined.

3 Global Tree Error Functions

The global tree optimization problem (GTO) is to find a feasible or almost feasible solution of the disjunctive inequalities corresponding to the decision tree. We propose several objectives which have the property that the minimum objective value is zero if and only if the solution is feasible for the disjunctive inequalities. If the disjunctive inequalities are not feasible, the problem minimizes some function of the classification error. For clarity we will begin with a discussion for the three-decision tree in Figure 2 and then extend the error functions to the more general case.

3.1 Sample Case

The basic idea is to calculate the error of each point and then minimize the sums of these errors. The error can be broken down into three parts: the error at each decision, the error at each leaf, and the total point error. At each decision, we look at the inequality that must be satisfied and then measure by how much it is violated. Assume the point must traverse the right path or "greater than" path at decision d. We want $A_i w^d - \gamma^d \geq 1$, so we define the error as $(y_g^d)_i^p = (-A_i w^d + \gamma^d + 1)_+$ where $(\nu)_+ = \max\{\nu, 0\}$. Note that a point at decision d can be of class $\mathcal{A}$ or $\mathcal{B}$, and it can traverse the left or right path. Thus, the four possible errors for a point (A_i or B_j) at decision d are:

$$\begin{array}{lcl}
A_i w^d-\gamma^d\geq 1 & \Rightarrow & (y_g^d)_i^p=(-A_i w^d+\gamma^d+1)_+\\
A_i w^d-\gamma^d\leq -1 & \Rightarrow & (y_l^d)_i^p=(A_i w^d-\gamma^d+1)_+\\
B_j w^d-\gamma^d\geq 1 & \Rightarrow & (z_g^d)_j^p=(-B_j w^d+\gamma^d+1)_+\\
B_j w^d-\gamma^d\leq -1 & \Rightarrow & (z_l^d)_j^p=(B_j w^d-\gamma^d+1)_+
\end{array}$$

The subscript l represents traversing the "less than" path at decision d and g represents traversing the "greater than" path at decision d. The error at each decision can be calculated to the power of p, where p is some positive integer. When $p = 1$, the absolute error is calculated. When $p = 2$, the squared error is calculated. In this paper we use $p = 1$.

The leaf error is a function of the decision errors along the path from the root to a leaf. We take the sum of the decision errors along each path to calculate the leaf error at a given leaf node. Thus for the tree in Figure 2:

$$\begin{array}{l} LeafErr(A_i, node\ 4) = ((y_l^1)_i^p\ +\ (y_l^2)_i^p) \\ LeafErr(A_i, node\ 6) = ((y_g^1)_i^p\ +\ (y_l^3)_i^p) \\ LeafErr(B_j, node\ 5) = ((z_l^1)_j^p\ +\ (z_g^2)_j^p) \\ LeafErr(B_j, node\ 7) = ((z_g^1)_j^p\ +\ (z_g^3)_j^p) \end{array} \tag{7}$$

We wish to minimize the leaf error for each point, so we define the point error as a function of the leaf errors. There are many possibilities of how we might define our objective to minimize the leaf error. One possible objective for minimizing the leaf error is to count the number of points misclassified. We call this the count error function:

$$\min_{y,z,w,\gamma} \begin{array}{l} \sum_{i=1}^{m_A} count(\min(LeafErr(A_i, node\ 4), LeafErr(A_i, node\ 6)))\ + \\ \sum_{j=1}^{m_B} count(\min(LeafErr(B_j, node\ 5), LeafErr(B_j, node\ 7))) \end{array} \tag{8}$$

where $count(\nu) := 1$ if $\nu > 0$ and $count(\nu) := 0$ if $\nu \leq 0$. The minimum of the leaf errors will be zero when a point is correctly classified by any one of the leaf nodes of that class. If all the points are correctly classified, this objective function will be zero. Otherwise, it will be the number of points misclassified.

Another possible objective for minimizing the leaf error is to use a winner-take-all approach of minimizing over the minimum leaf error for each point. We call this the minimum error function:

$$\min_{y,z,w,\gamma} \begin{array}{l} \sum_{i=1}^{m_A} \min(LeafErr(A_i, node\ 4), LeafErr(A_i, node\ 6))\ + \\ \sum_{j=1}^{m_B} \min(LeafErr(B_j, node\ 5), LeafErr(B_j, node\ 7)) \end{array} \tag{9}$$

For a point correctly classified, the minimum of the leaf error for that class will be zero. If all the points are classified correctly, this objective function will be zero. Otherwise, it will be the sum of the smallest leaf errors for each class.

A third possible objective is to minimize the product of the leaf errors for each class. We call this the product error function:

$$\min_{y,z,w,\gamma} \begin{array}{l} \sum_{i=1}^{m_A} (LeafErr(A_i, node\ 4) \cdot LeafErr(A_i, node\ 6))\ + \\ \sum_{j=1}^{m_B} (LeafErr(B_j, node\ 5) \cdot LeafErr(B_j, node\ 7)) \end{array} \tag{10}$$

This function also has a zero objective function value when all the points are correctly classified.

Each of the above problems is challenging to solve. The problems are nonconvex and have many local minima. Only the product error function with $p = 2$ is differentiable. The minimum error and count error functions are not even continuous.

To simplify the objective, we eliminate the plus functions by introducing linear constraints. We define y and z as nonnegative slack variables that are bounded below by the leaf error. The objective is then minimized over the resulting polyhedral region. The product error problem (10) becomes the following math programming problem:

$$\begin{array}{rlll}
\min\limits_{y,z,w,\gamma} & \multicolumn{3}{l}{\sum_{i=1}^{m_A}(LeafErr(A_i, node\ 4)\cdot LeafErr(A_i, node\ 6))\ +} \\
& \multicolumn{3}{l}{\sum_{j=1}^{m_B}(LeafErr(B_j, node\ 5)\cdot LeafErr(B_j, node\ 7))} \\
subject\ to & (y_l^1)_i \geq\ A_iw^1-\gamma^1+1 & (y_g^1)_i \geq\ -A_iw^1+\gamma^1+1 & \\
& (y_l^2)_i \geq\ A_iw^2-\gamma^2+1 & (y_l^3)_i \geq\ A_iw^3-\gamma^3+1 & \\
& (z_l^1)_j \geq\ B_jw^1-\gamma^1+1 & (z_g^1)_j \geq\ -B_jw^1+\gamma^1+1 & \\
& (z_g^2)_j \geq\ -B_jw^2+\gamma^2+1 & (z_g^3)_j \geq\ -B_jw^3+\gamma^3+1 & \\
& i=1,\ldots,m_A \quad\quad j=1,\ldots,m_B & y,z\ \geq\ 0 &
\end{array} \tag{11}$$

Similar transformations can be done for the count error problem and the minimum error problem. We will now formulate these problems for a general tree of fixed structure.

3.2 General Case

The preceding derivation of the decision, leaf, and total point errors can be extended to any multivariate binary decision tree. As was done for the general case in Section 2.2, let the tree have D decision nodes and $D+1$ leaf nodes. Assume the decision nodes are numbered 1 through D, and the leaf nodes are numbered $D+1$ through $2D+1$. Let θ_A be the set of leaf nodes classified as $\mathcal{A}$ and let θ_B be the set of leaf nodes classified as $\mathcal{B}$. Consider the path traversed from the root to a leaf node. For each leaf node k, define G^k as the index set of the decisions in which the right or "greater than" branch is traversed to reach leaf k. For each leaf node k, define L^k as the index set of the decisions in which the left or "less than" branch is traversed to reach leaf k.

The four possible errors at each decision d are still:

$$\begin{array}{lcl}
A_iw^d-\gamma^d \geq 1 & \Rightarrow & (y_g^d)_i^p = (-A_iw^d+\gamma^d+1)_+ \\
A_iw^d-\gamma^d \leq -1 & \Rightarrow & (y_l^d)_i^p = (A_iw^d-\gamma^d+1)_+ \\
B_jw^d-\gamma^d \geq 1 & \Rightarrow & (z_g^d)_j^p = (-B_jw^d+\gamma^d+1)_+ \\
B_jw^d-\gamma^d \leq -1 & \Rightarrow & (z_l^d)_j^p = (B_jw^d-\gamma^d+1)_+
\end{array} \tag{12}$$

where p is some positive integer. Recall, that when $p=1$ the absolute error is calculated and when $p=2$ the squared error is calculated. In this paper we use $p=1$. The subscript l represents traversing the "less than" path at decision d and g represents traversing the "greater than" path at decision d.

The leaf error is the sum of the decision errors along the path from the root to a leaf. Using the notation introduced in (12), the leaf error can be defined as:

Definition 3.1 (Leaf Error) *The leaf error of a point i in class $\mathcal{A}$ at leaf node k is defined by*

$$LeafErr(A_i, node\ k)\ =\ \sum_{d\in G^t}(y_g^d)_i^p + \sum_{d\in L^t}(y_l^d)_i^p \tag{13}$$

where $(y_g^d)_i^p = (-A_i w^d + \gamma^d + 1)_+$ *and* $(y_l^d)_i^p = (A_i w^d - \gamma^d + 1)_+$*, and* p *is some positive integer. The leaf error of a point* j *in class* $\mathcal{B}$ *at leaf node* k *is defined by*

$$LeafErr(B_j, node\ k) \;=\; \textstyle\sum_{d\in G^t}(z_g^d)_j^p + \sum_{d\in L^t}(z_l^d)_j^p \tag{14}$$

where $(z_g^d)_j^p = (-B_j w^d + \gamma^d + 1)_+$ *and* $(z_l^d)_j^p = (B_j w^d - \gamma^d + 1)_+$*, where* p *is some positive integer.*

Note that the leaf error for each point has as many summands as the depth of that leaf in the tree.

Once again, the plus function can be eliminated by introducing linear constraints. The objective is then minimized over the resulting polyhedral region. Counting the number of points misclassified becomes the following optimization problem:

Definition 3.2 (Count Error Problem) *For a fixed tree structure, the count error objective function which counts the number of points misclassified is:*

$$\begin{aligned} g(y,z,w,\gamma) \;=\; \min_{y,z,w,\gamma} \;& \textstyle\sum_{i=1}^{m_A} count(\min_{k\in\theta_A}(LeafErr(A_i, node\ k))) \;+ \\ & \textstyle\sum_{j=1}^{m_B} count(\min_{k\in\theta_B}(LeafErr(B_j, node\ k))) \end{aligned} \tag{15}$$

where $count(\nu) := 1$ *if* $\nu > 0$, $count(\nu) := 0$ *if* $\nu \le 0$*, and the leaf error is defined by Definition (3.1). The equivalent nonlinear programming problem is:*

$$\begin{aligned} g(y,z,w,\gamma) \;=\; \min_{y,z,w,\gamma} \;& \textstyle\sum_{i=1}^{m_A} count(\min_{k\in\theta_A}(LeafErr(A_i, node\ k))) \;+ \\ & \textstyle\sum_{j=1}^{m_B} count(\min_{k\in\theta_B}(LeafErr(B_j, node\ k))) \\ \textit{subject to}\;\; & (y_l^d)_i \ge\; A_i w^d - \gamma^d + 1 \\ & (y_g^d)_i \ge\; -A_i w^d + \gamma^d + 1 \\ & (z_l^d)_j \ge\; B_j w^d - \gamma^d + 1 \\ & (z_g^d)_j \ge\; -B_j w^d + \gamma^d + 1 \\ & i = 1,\ldots,m_A \quad j = 1,\ldots,m_B \\ & d = 1,\ldots,D \quad y,z \;\ge\; 0 \end{aligned} \tag{16}$$

If a tree exists of the given structure that correctly classifies all the points, then the problem has an optimal solution of zero. The same can be said of the formulations for the minimum error and the product error problems. The minimum error problem can be written as follows:

Definition 3.3 (Minimum Error Problem) *For a fixed tree structure, the minimum error objective function which minimizes the smallest leaf error for each point is given by:*

$$\begin{aligned} g(y,z,w,\gamma) \;=\; \min_{y,z,w,\gamma} \;& \textstyle\sum_{i=1}^{m_A} \min_{k\in\theta_A}(LeafErr(A_i, node\ k)) \;+ \\ & \textstyle\sum_{j=1}^{m_B} \min_{k\in\theta_B}(LeafErr(B_j, node\ k)) \end{aligned} \tag{17}$$

where the leaf error is defined by Definition (3.1). The equivalent nonlinear programming problem is:

$$\begin{array}{rrl} g(y,z,w,\gamma) & = \min\limits_{y,z,w,\gamma} & \sum_{i=1}^{m_A} \min_{k\in\theta_A}(LeafErr(A_i, node\ k)) + \\ & & \sum_{j=1}^{m_B} \min_{k\in\theta_B}(LeafErr(B_j, node\ k)) \\ & subject\ to & (y_l^d)_i \geq \ A_i w^d - \gamma^d + 1 \\ & & (y_g^d)_i \geq \ - A_i w^d + \gamma^d + 1 \\ & & (y_l^d)_j \geq \ B_j w^d - \gamma^d + 1 \\ & & (y_g^d)_j \geq \ - B_j w^d + \gamma^d + 1 \\ & & i = 1,\ldots,m_A \quad j = 1,\ldots,m_B \\ & & d = 1,\ldots,D \quad y,z \ \geq \ 0 \end{array} \tag{18}$$

The previous two problems are not continuous. The product error problem produces a continuous objective function that is differentiable. The product error problem can be written as:

Definition 3.4 (Product Error Problem) *For a fixed tree structure, the product error objective function, which is minimized over the sum of the products of the leaf errors, for each leaf node is given by:*

$$\begin{array}{rrl} g(y,z,w,\gamma) & = \min\limits_{y,z,w,\gamma} & \sum_{i=1}^{m_A}(\prod_{k\in\theta_A}(LeafErr(A_i, node\ k))) + \\ & & \sum_{j=1}^{m_B}(\prod_{k\in\theta_B}(LeafErr(B_j, node\ k))) \end{array} \tag{19}$$

where the leaf error is defined by Definition (3.1). The equivalent nonlinear programming problem is:

$$\begin{array}{rrl} g(y,z,w,\gamma) & = \min\limits_{y,z,w,\gamma} & \sum_{i=1}^{m_A}(\prod_{k\in\theta_A}(LeafErr(A_i, node\ k))) + \\ & & \sum_{j=1}^{m_B}(\prod_{k\in\theta_B}(LeafErr(B_j, node\ k))) \\ & subject\ to & (y_l^d)_i \geq \ A_i w^d - \gamma^d + 1 \\ & & (y_g^d)_i \geq \ - A_i w^d + \gamma^d + 1 \\ & & (y_l^d)_j \geq \ B_j w^d - \gamma^d + 1 \\ & & (y_g^d)_j \geq \ - B_j w^d + \gamma^d + 1 \\ & & i = 1,\ldots,m_A \quad j = 1,\ldots,m_B \\ & & d = 1,\ldots,D \quad y,z \ \geq \ 0 \end{array} \tag{20}$$

We have defined three possible error functions to minimize the error for a general tree of fixed structure. Each of these problems is NP-complete [17, 12, 21]. Although the constraints are linear, the objectives are nonconvex. We now justify why examining extreme point solutions to any of these problems is satisfactory.

4 Extreme Point Solutions

In this section, we will show it is sufficient to consider the extreme point solutions for each of the different error functions introduced in the last section.

Theorem 4.1 (Existence of Better Extreme Point Solutions) *Let $g(\bar{w}, \bar{\gamma}, \bar{y}, \bar{z})$ be the objective value for a feasible solution $\bar{w}, \bar{\gamma}, \bar{y}, \bar{z}$ of the count error problem (16). Then there exists a corresponding extreme point solution $\tilde{w}, \tilde{\gamma}, \tilde{y}, \tilde{z}$ such that $g(\tilde{w}, \tilde{\gamma}, \tilde{y}, \tilde{z}) \leq g(\bar{w}, \bar{\gamma}, \bar{y}, \bar{z})$. Equivalently, the number of points misclassified by the extreme point solution is less than or equal to the number of points misclassified by its corresponding feasible solution.*

Proof. Consider any feasible solution $\bar{w}, \bar{\gamma}, \bar{y}, \bar{z}$ of the count error problem (16). If it is an extreme point solution, we are done. So assume it is not an extreme point solution.

By Definition 3.1, a point i of class $\mathcal{A}$ is correctly classified if $LeafErr(A_i, node\ k) = 0$ for at least one leaf node k of class $\mathcal{A}$. This, along with the constraint $y \geq 0$, implies $(y_g^d)_i = 0$ for all $d \in G^k$ and $(y_l^d)_i = 0$ for all $d \in L^k$ for all leaf nodes k which make the leaf error zero. Similarly, a point j of class $\mathcal{B}$ is correctly classified if $LeafErr(B_j, node\ k) = 0$ for at least one leaf node k of class $\mathcal{B}$. This, along with the constraint $z \geq 0$, implies $(z_g^d)_j = 0$ for all $d \in G^k$ and $(z_l^d)_j = 0$ for all $d \in L^k$ for all leaf nodes k which make the leaf error zero.

For each point correctly classified by the feasible solution $\bar{w}, \bar{\gamma}, \bar{y}, \bar{z}$, find all the y and z that are zero by the construction just described, and call this index set Q. We can then form the following linear problem (LP) that by construction is known to have an optimal objective value of zero.

$$\begin{array}{rl}
\min\limits_{y,z,w,\gamma} & \sum_{q\in Q}((y_l^d)_i)_q + \sum_{q\in Q}((y_g^d)_i)_q + \\
& \sum_{q\in Q}((z_l^d)_j)_q + \sum_{q\in Q}((z_g^d)_j)_q \\
subject\ to & (y_l^d)_i \geq A_i w^d - \gamma^d + 1 \\
& (y_g^d)_i \geq -A_i w^d + \gamma^d + 1 \\
& (z_l^d)_j \geq B_j w^d - \gamma^d + 1 \\
& (z_g^d)_j \geq -B_j w^d + \gamma^d + 1 \\
& i = 1,\ldots,m_A \quad j = 1,\ldots,m_B \\
& d = 1,\ldots,D \quad y,z \geq 0
\end{array} \tag{21}$$

This LP can be put into standard form and solved using any LP algorithm such as the Simplex Method [28]. There exists an optimal basic feasible solution $\tilde{w}, \tilde{\gamma}, \tilde{y}, \tilde{z}$ with an objective value of zero. This solution is an extreme point. Since both the count error problem (16) and the LP have the same constraints, $\tilde{w}, \tilde{\gamma}, \tilde{y}, \tilde{z}$ is also an extreme point solution to the count error problem (16).

Now we will show that the count objective function value with the extreme point solution $\tilde{w}, \tilde{\gamma}, \tilde{y}, \tilde{z}$ is as good as or better than the count objective function value with the solution $\bar{w}, \bar{\gamma}, \bar{y}, \bar{z}$. Recall the count objective function:

$$g(y,z,w,\gamma) = \min_{y,z,w,\gamma} \begin{array}{l} \sum_{i=1}^{m_A} count(\min_{k\in\theta_A}(LeafErr(A_i, node\ k))) + \\ \sum_{j=1}^{m_B} count(\min_{k\in\theta_B}(LeafErr(B_j, node\ k))) \end{array} \tag{22}$$

where *count*$(\nu) := 1$ if $\nu > 0$ and *count*$(\nu) := 0$ if $\nu \leq 0$. Assume there are M points misclassified and N points correctly classified by the feasible solution $\bar{w}, \bar{\gamma}, \bar{y}, \bar{z}$ to the count error problem (16), where $M + N = m_A + m_B$. For each of the N points correctly classified by the feasible solution $\bar{w}, \bar{\gamma}, \bar{y}, \bar{z}$, the extreme point solution $\tilde{w}, \tilde{\gamma}, \tilde{y}, \tilde{z}$ will have those same N points classified correctly by the construction of the extreme point solution from the feasible solution. We optimized the LP over the y and z variables of the index set Q which were known to be zero valued in the feasible solution $\bar{w}, \bar{\gamma}, \bar{y}, \bar{z}$. Thus, these same y and z variables of the index set Q are zero in the optimal extreme point solution to the LP. The N points correctly classified by the feasible solution $\bar{w}, \bar{\gamma}, \bar{y}, \bar{z}$ to the count error problem (16) will also be correctly classified by the extreme point solution $\tilde{w}, \tilde{\gamma}, \tilde{y}, \tilde{z}$ of the LP. Thus the extreme point solution will have at least as many points classified correctly as the feasible solution $\bar{w}, \bar{\gamma}, \bar{y}, \bar{z}$, or equivalently $g(\tilde{w}, \tilde{\gamma}, \tilde{y}, \tilde{z}) \leq g(\bar{w}, \bar{\gamma}, \bar{y}, \bar{z})$. □

The following corollaries follow directly from the proof of Theorem 4.1.

Corollary 4.1 (Existence of Extreme Point Solution for the Count Error Problem) *The count error problem (16) has an extreme point optimal solution.*

If the points are separable by a tree with a given structure, then the minimum error and product error problems also have optimal extreme point solutions.

Corollary 4.2 (Zero Objective Value Extreme Point Solution to the Minimum Error Problem) *If the optimal objective value of the minimum error problem (18) is zero, then the problem has an extreme point solution.*

Proof. Note that an optimal objective value of zero for the minimum error problem (18) implies that all the points are classified correctly, or no points are misclassified. Thus, it follows directly from the proof of Theorem 4.1 that there exists an extreme point solution to the problem. □

Corollary 4.3 (Zero Objective Value Extreme Point Solution to the Product Error Problem) *If the optimal objective value of the product error problem (20) is zero, then the problem has an extreme point solution.*

When the points are not separable by the given tree structure, then the minimum error or product error problems may not have an optimal extreme point solution. Using Theorem 4.1, we know there exists an extreme point that, in terms of the number of points misclassified, is as good or better than any optimal solution or the minimum error or product error problems.

Corollary 4.4 (Better Extreme Point Exists for Minimum Error Problem) *For any optimal solution, $\bar{w}, \bar{\gamma}, \bar{y}, \bar{z}$, of the minimum error problem (18), there exists an extreme point solution, $\tilde{w}, \tilde{\gamma}, \tilde{y}, \tilde{z}$, such that the number of points misclassified by the extreme point solution is less than or equal to the number of points misclassified by its corresponding feasible solution.*

Corollary 4.5 (Better Extreme Point Exists for Product Error Problem) *For any optimal solution, $\bar{w}, \bar{\gamma}, \bar{y}, \bar{z}$, of the product error problem (20), there exists an extreme point solution, $\tilde{w}, \tilde{\gamma}, \tilde{y}, \tilde{z}$, such that the number of points misclassified by the extreme point solution is less than or equal to the number of points misclassified by its corresponding feasible solution.*

This quality of possessing extreme point solutions suggests that these problems can be solved using algorithms that only traverse the vertices. There are finitely many extreme points in the polyhedral region as compared to the infinite number of values possible for w and γ. In the next two sections, we describe two algorithms for exploring the vertices: a Frank-Wolfe Algorithm and an Extreme Point Tabu Search Algorithm.

5 Optimization Algorithms For GTO

Specialized algorithms have been developed for searching the extreme points of polyhedral regions. We can take advantage of the numerically efficient methods for representing extreme points developed for the Simplex Method of linear programming [27]. In the Frank-Wolfe algorithm (FW), nonlinear problems are solved using a series of linear programs created by linearizing the objective function. A Frank-Wolfe method was successfully applied to the product error problem [7, 3]. However, the Frank-Wolfe algorithm is a descent method that stops at the first local minimum encountered and is limited to differentiable functions. We would like a more robust approach that can be used for nondifferentiable objectives, such as the count objective. Thus, we have developed an extreme point tabu search (EPTS) algorithm. We begin with a brief review of the FW algorithm and then discuss EPTS.

5.1 Frank-Wolfe Algorithm

A Frank-Wolfe algorithm can be applied to the product error problem (20). Consider the more general problem of minimizing $f(x)$ over the convex set $x \in \mathcal{X}$ where f is differentiable. The Frank-Wolfe algorithm for this problem is:

Algorithm 5.1 (Frank-Wolfe algorithm) *Start with any $x^0 \in \mathcal{X}$. Compute x^{i+1} from x^i as follows.*

- $$v^i \in arg\ vertex \min_{x \in \mathcal{X}} \ \nabla f(x^i)x$$
- $$Stop\ if \ \ \nabla f(x^i)v^i = \nabla f(x^i)x^i$$
- $$x^{i+1} = (1-\lambda^i)x^i + \lambda^i v^i \ where$$
  $$\lambda^i \in arg \min_{0 \le \lambda \le 1} f((1-\lambda)x^i + \lambda v^i)$$

Applied to the product error problem, this algorithm terminates at a point satisfying the following minimum principle necessary optimality condition: $\nabla f(x^j)(x - x^j) \geq 0$, for all $x \in \mathcal{X}$, or each accumulation point $\bar{x}$ of the sequence $\{x^i\}$ satisfies the minimum principle. Since FW gets stuck at local minima and does not apply to nondifferentiable functions, we also developed a heuristic search approach.

5.2 Extreme Point Tabu Search

We have developed an Extreme Point Tabu Search (EPTS) algorithm to solve the GTO problem.[1] Tabu Search (TS) is a heuristic that has achieved great success on a wide variety of practical optimization problems [14]. Extreme point tabu search has been previously applied to other extreme point tabu search problems [34, 8]. A very basic TS search algorithm is described below. For each point in the search space a "neighborhood" is defined. The algorithm moves to the best neighbor of the current solution that is not "tabu". If the best neighbor is tabu and it exceeds some "aspiration criteria", then a move may be taken even though it is tabu. The dynamically changing tabu lists and the neighborhoods help this nonmonotonic algorithm move out of a local minimum and continue searching for better solutions without cycling.

```
Basic Tabu Search
1.  Find an initial starting point.
2.  If iteration limit is exceeded, stop.
3.  Until a move is chosen:
        Find best neighbor.
        If not tabu, take that move.
        Else if tabu and aspiration criteria met, take that move.
        Else consider next best neighbor as a possible move.
4.  Go to step 1
```

Our extreme point tabu search (EPTS) is an extension of the algorithm described above. Other versions of tabu search applied to extreme points can be found in [16, 1, 20]. We begin with a description of the features of our algorithm. The algorithm utilizes both recency-based and frequency-based memory and oscillates between local improvement and diversification phases. The resulting algorithm is summarized in Algorithm 5.2.

EPTS is a tabu search on the extreme points of an optimization problem with linear constraints of the following general form:

$$\min_{x} \ f(x), \ \textit{subject to } Ax = b, \ x \geq 0 \tag{23}$$

where $x \in R^c$ and A is an $m \times c$ matrix. Standard procedures exist for transforming problems with linear inequalities and/or unconstrained variables into this format. The objective function of (23) need not be linear, continuous, or even differentiable. The polyhedron formed from the linear constraints has $\frac{n!}{m!(n-m)!}$ extreme points. Each extreme point of the polyhedral region formed by the constraints of (23) corresponds to a basic feasible solution. A basic feasible solution consists of m variables that form a basis. The remaining nonbasic variables are set to zero. Once a nonbasic variable is selected to become basic, the ratio test is used to select the basic variable to move out of the basis. A pivot swaps the incoming and outgoing variables.

For EPTS, the neighbors are the adjacent extreme points as defined by the nonbasic variables. Pivoting in one of these variables will move from one extreme point to a neighboring extreme point. The initial starting point is any basic feasible solution of problem (23).

[1] Many thanks to Fred Glover of the University of Colorado for suggesting Tabu Search for this problem.

The best neighbor is based upon the move with the smallest objective function value. Since the number of adjacent extreme points and cost of function evaluation can be quite large, a candidate list is used to limit the number of function evaluations performed when selecting a move. The candidate list consists of all the improving pivots identified by the entering variable at some iteration. Possible moves from the candidate list are evaluated until a non-tabu improving move is found or the aspiration criterion is satisfied. The selected move is then removed from the candidate list. If no improving non-tabu moves are found then the candidate list is remade. If still no improving moves are found then a tabu move is taken. The candidate list is also remade if it becomes less than 1/8 its original length.

Since the choice of variables in the basis uniquely determines the extreme point, the memory structures need only track which variables are in the basis. For recency-based memory, the iteration number is recorded whenever a variable enters or exits the basis. For frequency-based memory, we keep count of the number of times each variable is used in the basis. We increment the count of the variables in the basis at every iteration, not just for critical events as suggested in [15]. We explored the critical event strategy but found it did not enhance the quality of our results. The best solution found so far in terms of the given objective function is used as the aspiration criterion. The number of iterations that a variable remains tabu is called the tabu tenure. Moves can become tabu in two ways. If a nonbasic variable is required to reenter the basis before its tabu tenure has expired, the move is tabu. A move may also be tabu if a basic variable is required to leave the basis before its tabu tenure has expired. The tabu tenure of nonbasic variables is longer than that of basic variables because the number of possible entering variables is much larger than that of exiting variables.

EPTS oscillates between a local improvement mode and a diversification mode. In the local improvement mode, the objective function is used to evaluate the quality of the move. The diversification strategy is a variation of the approach suggested in [15]. The following frequency penalty objective function is used to force the algorithm into unexplored areas of the search space:

$$penalty(x) \;=\; f(x) + \frac{\rho}{iteration} \sum_{i \in basis} frequency(i) \tag{24}$$

where i belongs to the set of indices of the variables in the current basis, $f(x)$ is the original objective function value, ρ is a large penalty constant, *iteration* is the number of the current iteration, and $frequency(i)$ is a count of the number of times the variable has appeared in the basis. Since we are only ranking adjacent extreme points at any iteration, we need only calculate

$$penalty(x) \;=\; f(x) + \frac{\rho}{iteration}(frequency(entering\ i) - frequency(exiting\ i)) + C \tag{25}$$

where *entering* i is the index of the entering variable, *exiting* i is the index of the departing variable and C is the sum of the frequencies of the variables in the old basis. In practice, a large constant can be used for C provided it is sufficiently large to avoid negative values of $penalty(x)$.

The algorithm starts in the local improvement mode using the original objective function. If the objective function values have not improved significantly in the last k iterations, the algorithm switches to the diversification mode. In this mode, the objective function values are obtained using $penalty(x)$. EPTS continues in diversification mode until a maximum number of diversification iterations is reached. We say the objective has not improved if it has not changed by $\delta\%$ in the last k iterations.

Algorithm 5.2 (Extreme Point Tabu Search)
Start with an initial basic feasible solution for Problem (23).

```
1. Start in local improvement mode with original objective.
2. If iteration limit is exceeded or solution is optimal, stop.
3. (a) If in local improvement mode and no progress is being made then switch to
       diversification mode with penalized frequency objective (25).
   (b) Else if in diversification mode and the maximum number of diversification steps
       has been reached, switch to local improvement mode with original objective.
4. Until a move is chosen
   (a) If the candidate list is too small or all the moves are tabu, then remake the
       candidate list.
   (b) If the candidate list was just made consider best move on the list, otherwise find
       first improving move in the candidate list.
   (c) If not tabu, take that move.
   (d) Else if tabu and aspiration criterion met, take that move.
   (e) Else if all moves are tabu and the candidate list has just been remade, take the
       best tabu move.
5. Perform move and update frequency and recency memories.
6. Go to step 2.
```

The EPTS algorithm is a Tabu Search on the extreme points of a polyhedron. Our algorithm can be applied to any problem that can be formulated as an optimization problem with linear constraints. The GTO problem is just one such problem to which EPTS applies.

6 Computational Results

To assess the effectiveness of Global Tree Optimization we performed two different computational experiments. In the first experiment we tried the three different error functions and two optimization methods for solving the GTO problem. Randomly generated data with known solutions were used so we could determine if the globally optimal solution was found. Based on the results of the first experiment, we applied the best GTO methods to a wider set of both randomly generated and real-world problems. In both algorithms, Quinlan's C4.5

[30] was used for a baseline comparison of how a very powerful and popular greedy univariate decision tree algorithm would perform on each problem. To keep as many factors constant as possible, a tree with three decisions such as the one in Figure 2 was used for GTO. The methods described in this paper are applicable to more general trees.

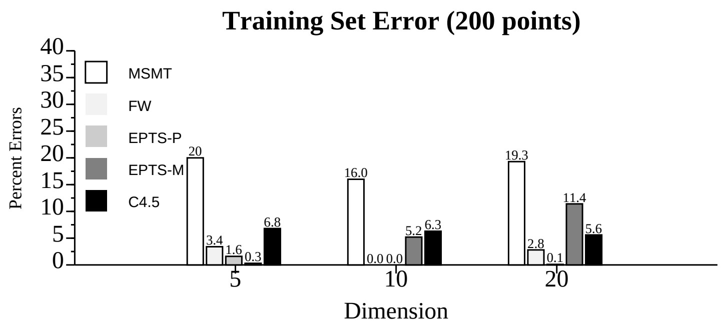
The random data was created by first generating the three decisions in a seven-node decision tree. Each weight and threshold were uniformly generated between -1 and 1. Then points were randomly generated in the unit cube and classified using the generated tree. A training set of 200 or 500 points was generated as well as a testing set of 2000 points.

Three different formulations of GTO were investigated: the product error problem (20), the minimum error problem (18) and the count error problem (16). The EPTS algorithm was used to optimize all three formulations. The FW algorithm could only be applied to the product error problem since the remaining problems are not differentiable. The results for the count error problem were uniformly bad. The EPTS algorithm with the count error function performs poorly since frequently moving to an adjacent extreme point does not change the number of points misclassified. Thus EPTS could not assess which moves are better. Thus no results are reported for the count error problem.

The linear programming package MINOS 5.4 [23] was used to implement both FW and EPTS. FW was implemented using the standard linear programming subroutines provided in MINOS. MINOS was customized so that EPTS could select the pivots to be taken. The following parameter setting for the EPTS algorithm were chosen through experimentation. We varied the parameters as a function of c, the number of variables in the problem. If the objective had not changed by 2% in $0.10 * c$ iterations, $0.15 * c$ frequency moves were taken. The tabu tenure for nonbasic variables was set to $\sqrt{c/8}$ and for basic variables to 1/4 the tabu tenure of the nonbasic variables. An iteration limit of 10,000 was used for each data set.

As a starting point for the EPTS and FW algorithms, an initial tree was generated using the greedy MSMT decision tree algorithm [2]. MSMT used a linear program [6] to recursively construct the decisions. The tree was then pruned to three decisions. Both EPTS and FW can be started using a random starting point, but we found the MSMT starting point worked better.

The results for randomly generated data of 5, 10 and 20 dimensions trained on 200 points and tested on 2000 points are given in Figure 4. The training set and testing set errors are reported. FW outperformed C4.5 on all the training and testing sets. EPTS outperformed FW, and thus also C4.5, on all the training and testing sets. C4.5 averaged 11.0, 16.4 and 22.2 decisions per tree for its solutions to the 5, 10, and 20 dimension problems, respectively. The results for both FW and EPTS algorithms used a GTO formulation fixed at 3 decisions. So not only did both FW and EPTS out perform C4.5 in training and testing set errors, they did so with only 3 multivariate decisions rather than 11 or more univariate decisions. The product error function optimized using EPTS performed the best of the all the methods tested. Note that no method consistently achieved 100% accuracy on any training set even though such a solution existed. It is not surprising that neither FW nor EPTS found the optimal solution since the underlying problem is NP-complete [12, 21]. The smallest GTO problem solved, 5 dimensions and 200 training points, has on the order of 10^{485} extreme points. The largest GTO problem, solved with 9 dimensions and 768 points, has on the



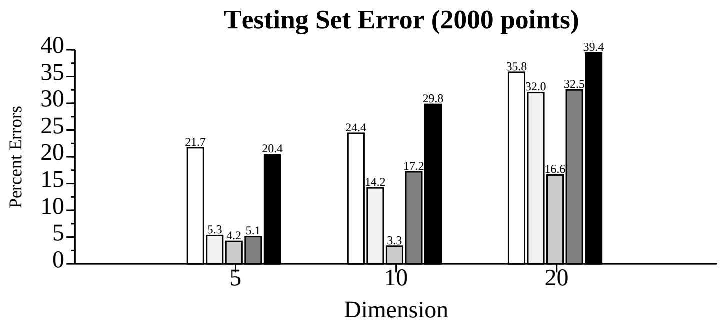


Figure 4: Comparison of three methods: FW, EPTS, and C4.5 on randomly generated problems. The MSMT starting point for FW and EPTS is also given. For EPTS, results are given for the minimum error (EPTS-M) and product error (EPTS-P) problems of a three-decision tree structure like that of Figure 2.

Average Training and Testing Set Error in Percent

		Number		Method			# Decisions	
Data Set	Dimension	of Points	Set	FW	EPTS	C4.5	C4.5	GTO
Generated	5	500	train	3.0	**2.4**	6.9	24.8	3.0
			test	4.1	**3.8**	19.4		
Generated	10	500	train	6.3	**3.6**	6.0	40.0	3.0
			test	10.0	**8.0**	30.0		
Generated	20	500	train	3.3	**2.5**	5.6	47.4	3.0
			test	**11.5**	12.2	36.7		
Heart	13	297	train	17.7	15.2	**12.0**	9.4	3.0
			test	**17.8**	20.6	22.2		
Cancer	9	682	train	**2.0**	2.3	6.3	6.0	3.0
			test	4.0	**3.4**	6.6		
Diabetes	8	768	train	24.1	22.0	**12.6**	29.6	3.0
			test	25.1	**23.2**	28.0		
Liver	6	345	train	28.1	35.3	**15.5**	19.4	3.0
			test	**32.2**	38.6	35.6		

Table 1: Comparison of classification error found by three methods: Frank-Wolfe, EPTS, and C4.5. The product error function (20) was used for FW and EPTS on a three-decision tree with structure as seen in Figure 2.

order of 10^{1485} extreme points.

In the second experiment, GTO was evaluated on larger random problems and real-world datasets. The real-world data[2] consisted of: the BUPA Liver Disease dataset (Liver); the PIMA Indians Diabetes dataset (Diabetes), the Wisconsin Breast Cancer Database (Cancer) [35], and the Cleveland Heart Disease Database (Heart) [13]. Five fold cross validation was used for the real-life problems. Each dataset was divided into 5 parts. Training sets used 4/5 of the data and testing was done on the remaining 1/5. This was repeated five times leaving out a different test set for each run. For the generated data sets, five pairs of training and testing data sets were generated for each case: 5, 10, and 20 dimensions, with 500 points per training set and 2000 points per testing set. The results were then averaged.

The results of the second experiment are summarized in Table 1. Since the minimum product error performed uniformly better in the first experiment, only results for the minimum product error optimized with FW and EPTS are shown. On every dataset, one of the GTO methods achieved better testing set accuracy than C4.5. Once again, trees used within GTO were dramatically smaller than those of C4.5 in terms of the number of decisions. By using multivariate decisions and a nongreedy approach, GTO produces trees that are arguably more interpretable than the trees produced by C4.5. The EPTS method performed better than FW on 4 of the 7 problems. This suggests that a hybrid approach

[2]All these datasets are available in the machine learning repository at the University of California at Irvine at http://www.ics.uci.edu/~mlearn/Machine-Learning.html.

combining both approaches may do even better. In fact this approach is investigated in [8]. Recall that a decision tree with three decisions was used for GTO on all the datasets. Other tree structures may better reflect the underlying structure of these problems and yield even better results for GTO.

7 Conclusions and Future Work

We have proposed a family of methods for formulating decision tree construction as an optimization problem. GTO can be used to optimize decision trees while maintaining their logical structure. Unlike greedy methods, GTO optimizes all the decisions within a multivariate decision tree concurrently. A given multivariate decision tree can be represented as a set of disjunctive linear inequalities. We characterized several objective functions that can be used to minimize the infeasibilities in the disjunctive linear inequalities, and thus the classification error as well. For all the error formulations, we proved that an extreme point solution exists that is as good as or better than the optimal solution in terms of the number of points misclassified. Two methods for searching the extreme points were examined: a Frank-Wolfe algorithm and a new extreme point tabu search approach. EPTS is a general purpose search algorithm that can be used on other problems for which extreme point solutions are desired. Unlike prior approaches, EPTS can be used for nondifferentiable and discontinuous objective function formulations. Computational results show that the GTO methods performed better than C4.5 on all 10 datasets attempted. Computationally, the GTO method with the product error function optimized using EPTS worked the best.

GTO has great potential as a technique for data mining and other practical classification problems. The techniques described here can be applied to any size decision tree. GTO can be used with greedy algorithms to construct and prune trees. It can be used for incorporating domain knowledge and transferring knowledge between applications. Since EPTS is a general purpose search algorithm, we can create new error functions that reflect application-specific criteria and the objectives of the ultimate user of the decision tree. In this paper we investigated a family of nonparametric error objective functions. However, GTO optimized with EPTS technique could be adapted to parametric formulations of the problem as well.

References

[1] R. Aboudi and K. Jőrnsten. Tabu search for general zero-one integer programs using the pivot and complement heuristic. *ORSA Journal on Computing*, 6(1):82–936, 1994.

[2] K. P. Bennett. Decision tree construction via linear programming. In M. Evans, editor, *Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society Conference*, pages 97–101, Utica, Illinois, 1992.

[3] K. P. Bennett. Global tree optimization: A non-greedy decision tree algorithm. *Computing Science and Statistics*, 26:156–160, 1994.

[4] K. P. Bennett. Optimal decision trees through multilinear programming. Manuscript, Rensselaer Polytechnic Institute, Troy, New York, 1995.

[5] K. P. Bennett and E. J. Bredensteiner. A parametric optimization method for machine learning. R.P.I. Math Report No. 217, Rensselaer Polytechnic Institute, Troy, New York, 1995. To appear in *INFORMS Journal on Computing.*

[6] K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

[7] K. P. Bennett and O. L. Mangasarian. Bilinear separation of two sets in n-space. *Computational Optimization and Applications*, 2:207–227, 1993.

[8] J. Blue and K. Bennett. Hybrid extreme point tabu search. R.P.I. Math Report No. 240, Rensselaer Polytechnic Institute, Troy, NY, 1996. To appear in the European Journal of Operations Research.

[9] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth International, California, 1984.

[10] C. E. Brodley and P. E. Utgoff. Multivariate decision trees. *Machine Learning*, 19(1):45–77, 1995.

[11] W. Buntine. Learning classification trees. In *In Artificial Intelligence Frontiers in Statistics: AI and Statistics III*, pages 183–201, London, 1993. Chapman and Hall.

[12] G. Das and M. G. Goodrich. On the complexity of optimization problems for 3-dimensional convex polyhedra and decision trees. In *Workshop on Algorithms and Data Structures (WADS)*, 1995.

[13] R. Detrano, A. Janosi, W. Steinbrunn, M. Pfisterer, J. Schmid, S. Sandhu, K. Guppy, S. Lee, and V. Froelicher. International application of a new probability algorithm for the diagnosis of coronary artery disease. *American Journal of Cardiology*, 64:304–310, 1989.

[14] F. Glover. Tabu search - part I. *ORSA Journal of Computing*, 1(3):190–206, 1989.

[15] F. Glover. Tabu search fundamentals and uses. Technical report, School of Business, University of Colorado, Boulder, Colorado, 1995.

[16] F. Glover and A. Løkketangen. Probabilistic tabu search for zero-one mixed integer programming problems. Manuscript, School of Business, University of Colorado, 1994.

[17] L. Hyafile and R. Rivest. Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5:1:15–17, 1976.

[18] M. Jordan. A statistical approach to decision tree modeling. In M. Warmuth, editor, *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, New York, 1994. ACM Press.

[19] M. Jordan and R. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(3):181–214, 1994.

[20] A. Løkketangen, K. Jőrnsten, and S. Storøy. Tabu search within a pivot and complement framework. *International Transactions of Operations Research*, 1(3):305–316, 1994.

[21] N. Megiddo. On the complexity of polyhedral separability. *Discrete and Computational Geometry*, 3:325–337, 1988.

[22] J. Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4:227–243, 1989.

[23] B.A. Murtagh and M.A. Saunders. MINOS 5.4 user's guide. Technical Report SOL 83.20, Stanford University, 1993.

[24] S. Murthy, S. Kasif, S. Salzberg, and R. Beigel. OC1: Randomized induction of oblique decision trees. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 322–327, Boston, MA, 1993. MIT Press.

[25] S. Murthy and S. Salzburg. Decision tree induction: How effective is the greedy heuristic? In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, 1995.

[26] S. Murthy and S. Salzburg. Lookahead and pathology in decision tree induction. In *Proceedings of of the Fourteenth Intl. Joint Conference. on Artificial Intelligence*, 1995.

[27] K. Murty. *Linear Programming.* Wiley, New York, 1983.

[28] K.G. Murty. *Linear Programming.* John Wiley & Sons, New York, New York, 1983.

[29] J. R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(1):221–234, 1987.

[30] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[31] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations. In D.E. Rumelhart and J.L. McClelland, editors, *Parallel Distributed Processing*, pages 318–362, Cambridge, Massachusetts, 1986. MIT Press.

[32] P. Smyth, A. Gray, and U. Fayyad. Retrofitting decision tree classifiers using kernel density estimation. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 506–514, Amherst, MA, 1995.

[33] W.N. Street. Cancer diagnosis and prognosis via linear-programming-based machine learning. Technical Report 94-14, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, August 1994. Ph.D. thesis.

[34] M. Sun and P. G. McKeown. Tabu search applied to the general fixed charge problem. *Annals of Operations Research*, 41:405–420, 1993.

[35] W. H. Wolberg and O.L. Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proceedings of the National Academy of Sciences,U.S.A.*, 87:9193–9196, 1990.