Global Tree Optimization: A Non-greedy Decision Tree Algorithm

Article · April 1995

Source: CiteSeer

CITATIONS
54

READS
1,494

1 author:

Kristin P. Bennett
Rensselaer Polytechnic Institute
149 PUBLICATIONS **6,784** CITATIONS

SEE PROFILE

Global Tree Optimization: A Non-greedy Decision Tree Algorithm

Kristin P. Bennett
Email bennek@rpi.edu
Department of Mathematical Sciences
Rensselaer Polytechnic Institute
Troy, NY 12180 *

Abstract

A non-greedy approach for constructing globally optimal multivariate decision trees with fixed structure is proposed. Previous greedy tree construction algorithms are locally optimal in that they optimize some splitting criterion at each decision node, typically one node at a time. In contrast, global tree optimization explicitly considers all decisions in the tree concurrently. An iterative linear programming algorithm is used to minimize the classification error of the entire tree. Global tree optimization can be used both to construct decision trees initially and to update existing decision trees. Encouraging computational experience is reported.

1 Introduction

Global Tree Optimization (GTO) is a new approach for constructing decision trees that classify two or more sets of n-dimensional points. The essential difference between this work and prior decision tree algorithms (e.g. CART [5] and ID3 [10]) is that GTO is non-greedy. For greedy algorithms, the "best" decision at each node is found by optimizing some splitting criterion. This process is started at the root and repeated recursively until all or almost all of the points are correctly classified. When the sets to be classified are disjoint, almost any greedy decision tree algorithm can construct a tree consistent with all the points, given a sufficient number of decision nodes. However, these trees may not generalize well (i.e., correctly classify future not-previously-seen points) due to over-fitting or over-parameterizing the problem. In practice decision nodes are pruned from the tree. Typically, the pruning process does not allow the remaining decision nodes to be adjusted, thus the tree may still be over-parameterized. The strength of the greedy algorithm is that by growing the tree and pruning it, the greedy algorithm determines the structure of the tree, the class at each of the leaves, and the decision at each non-leaf node. The limitations of greedy approaches are that locally "good" decisions may result in a bad overall tree and existing trees are difficult to update and modify.

GTO overcomes these limitations by treating the decision tree as a function and optimizing the classification error of the entire tree. The function is similar to the one proposed for MARS [8], however MARS is still a greedy algorithm. Greedy algorithms optimize one node at a time and then fix the resulting decisions. GTO starts from an existing tree. The structure of the starting tree (i.e. the number of decisions, the depth of the tree, and the classification of the leaves) determines the classification error function. GTO minimizes the classification error by changing all the decisions concurrently while keeping the underlying structure of the tree fixed. The advantages of this approach over greedy methods are that fixing the structure helps prevent overfitting or overparameterizing the problem, locally bad but globally good decisions can be made, existing trees can be re-optimized with additional data, and domain knowledge can be more readily applied. Since GTO requires the structure of the tree as input, it complements (not replaces) existing greedy decision tree methods. By complementing greedy algorithms, GTO offers the promise of making decision trees a more powerful, flexible, accurate, and widely accepted paradigm.

Minimizing the global error of a decision tree with fixed structure is a non-convex optimization problem. The problem of constructing a decision tree with a fixed number of decisions to correctly classify two or more sets is a special case of the NP-complete polyhedral separability problem [9]. Consider this seemingly simple but NP-complete problem [9]: Can a tree with just two decision nodes correctly classify two disjoint point sets? In [4],

*This material is based on research supported by National Science Foundation Grant 949427. This paper appeared in *Computing Science and Statistics*, 26, pg. 156-160, 1994.

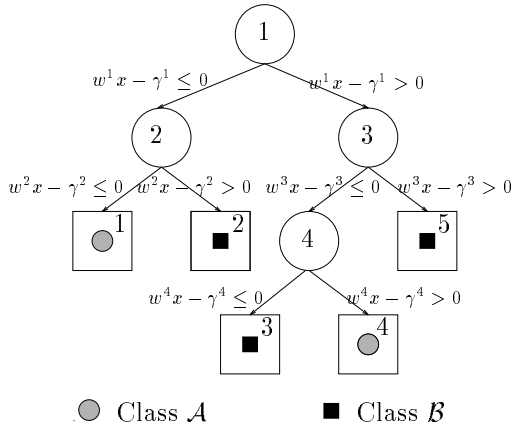
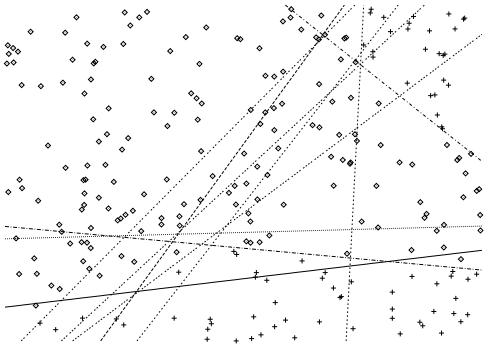


Figure 1: A typical two-class decision tree

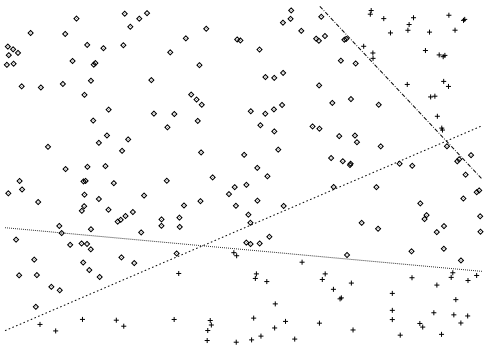
this problem was formulated as a bilinear program. We now extend this work to general decision trees, resulting in a multilinear program that can be solved using the Frank-Wolfe algorithm proposed for the bilinear case.

This paper is organized as follows. We begin with a brief review of the well-known case of optimizing a tree consisting of a single decision. The tree is represented as a system of linear inequalities and the system is solved using linear programming. In Section 3 we show how more general decision trees can be expressed as a system of disjunctive linear inequalities and formulated as a multilinear programming problem. Section 4 explains the iterative linear programming algorithm for optimizing the resulting problem. Computational results and conclusions are given in Section 5.

GTO applies to binary trees with a multivariate decision at each node of the following form: If x is a point being classified, then at decision node d, if $xw^d > \gamma^d$ the point follows the right branch, if $xw^d \leq \gamma^d$ then the point follows the left branch. The choice of which branch the point follows at equality is arbitrary. This type of decision has been used in greedy algorithms [6, 1]. The univariate decisions found by CART [5] for continuous variables can be considered special cases of this type of decision with only one nonzero component of w. A point is classified by following the path of the point through the tree until it reaches a leaf node. A point is strictly classified by the tree if it reaches a leaf of the correct class and equality does not hold at any decision along the path to the leaf (i.e. $xw^d \neq \gamma^d$ for any decision d in the path). Although GTO is applicable to problems with many classes, for simplicity we limit discussion to the problem of classifying the two sets $\mathcal{A}$ and $\mathcal{B}$. A sample of such a tree is given in Figure 1. Let $\mathcal{A}$ consist of k points contained in R^n and $\mathcal{B}$ consist of m points contained in R^n. Let A_j denote the jth point in $\mathcal{A}$.



(a) Tree found by Greedy LP Algorithm



(b) Tree found by GTO

Figure 2: Geometric depiction of decision trees

2 Optimizing a Single Decision

Many methods exist for minimizing the error of a tree consisting of a single decision node. We briefly review one approach which formulates the problem as a set of linear inequalities and then uses linear programming to minimize the errors in the inequalities [3]. The reader is referred to [3] for full details of the practical and theoretical benefits of this approach.

Let $xw = \gamma$ be the plane formed by the decision. For any point x, if $xw < \gamma$ then the point is classified in class $\mathcal{A}$, and if $xw > \gamma$ then the point is classified in class $\mathcal{B}$. If $xw = \gamma$ the class can be chosen arbitrarily. All the points in $\mathcal{A}$ and $\mathcal{B}$ are strictly classified if there exist w and γ

such that

$$\begin{array}{ll} A_j w - \gamma < 0 & j = 1 \ldots m \\ B_i w - \gamma > 0 & i = 1 \ldots k \end{array} \quad (1)$$

or equivalently

$$\begin{array}{ll} -A_j w + \gamma \geq 1 & j = 1 \ldots m \\ B_i w - \gamma \geq 1 & i = 1 \ldots k \end{array} \quad (2)$$

Note that Equations (1) and (2) are alternative definitions of linear separability. The choice of the constant 1 is arbitrary. Any positive constant may be used.

If $\mathcal{A}$ and $\mathcal{B}$ are linearly separable then Equation (2) is feasible, and the linear program (LP) (3) will have a zero minimum. The resulting (w, γ) forms a decision that strictly separates $\mathcal{A}$ and $\mathcal{B}$. If Equation (2) is not feasible, then LP (3) minimizes the average misclassification error within each class.

$$\begin{array}{lll} \min\limits_{w,\gamma} & \frac{1}{m}\sum\limits_{j=1}^{m} y_j + \frac{1}{k}\sum\limits_{i=1}^{k} z_i & \\ s.t. & y_j \geq A_j w - \gamma + 1 \quad y_j \geq 0 & j = 1 \ldots m \\ & z_i \geq -B_i w + \gamma + 1 \quad z_i \geq 0 & i = 1 \ldots k \end{array} \quad (3)$$

LP (3) has been used recursively in a greedy decision tree algorithm called Multisurface Method-Tree (MSMT) [1]. While it compares favorably with other greedy decision tree algorithms, it also suffers the problem of all greedy approaches. Locally good but globally poor decisions near the root of the tree can result in overly large trees with poor generalization. Figure 2 shows an example of a case where this phenomenon occurs. Figure 2a depicts the 11 planes used by MSMT to completely classify all the points. The decisions chosen near the root of the tree are largely redundant. As a result the decisions near the leaves of the tree are based on an unnecessarily small number of points. MSMT constructed an excessively large tree that does not reflect the underlying structure of the problem. In contrast, GTO was able to completely classify all the points using only three decisions (Figure 2b).

3 Problem Formulation

For general decision trees, the tree can be represented as a set of disjunctive inequalities. A multilinear program is used to minimize the error of the disjunctive linear inequalities. We now consider the problem of optimizing a tree with the structure given in Figure 1, and then briefly consider the problem for more general trees.

Recall that a point is strictly classified by the tree in Figure 1 if the point reaches a leaf of the correct classification and equality does not hold for any of the decisions along the path to the leaf. A point $A_j \in \mathcal{A}$ is strictly classified if it follows the path through the tree to the first or fourth leaf node, i.e. if

$$\left\langle \begin{array}{r} A_j w^1 - \gamma^1 + 1 \leq 0 \\ A_j w^2 - \gamma^2 + 1 \leq 0 \end{array} \right\rangle$$
$$\textbf{or}$$
$$\left\langle \begin{array}{r} -A_j w^1 + \gamma^1 + 1 \leq 0 \\ A_j w^3 - \gamma^3 + 1 \leq 0 \\ -A_j w^4 + \gamma^4 + 1 \leq 0 \end{array} \right\rangle \quad (4)$$

or equivalently

$$\begin{array}{c} (A_j w^1 - \gamma^1 + 1)_+ \cdot (A_j w^2 - \gamma^2 + 1)_+ = 0 \\ \textbf{or} \\ (-A_j w^1 + \gamma^1 + 1)_+ \cdot (A_j w^3 - \gamma^3 + 1)_+ \cdot \\ (-A_j w^4 + \gamma^4 + 1)_+ = 0 \end{array} \quad (5)$$

where $(\zeta)_+ := \max\{\zeta, 0\}$.

Similarly a point $B_i \in \mathcal{B}$ is strictly classified if it follows the path through the tree to the second, third, or fifth leaf node, i.e. if

$$\left\langle \begin{array}{r} B_i w^1 - \gamma^1 + 1 \leq 0 \\ -B_i w^2 + \gamma^2 + 1 \leq 0 \end{array} \right\rangle$$
$$\textbf{or}$$
$$\left\langle \begin{array}{r} -B_i w^1 + \gamma^1 + 1 \leq 0 \\ B_i w^3 - \gamma^3 + 1 \leq 0 \\ B_i w^4 - \gamma^4 + 1 \leq 0 \end{array} \right\rangle \quad (6)$$

$$\textbf{or}$$
$$\left\langle \begin{array}{r} -B_i w^1 + \gamma^1 + 1 \leq 0 \\ -B_i w^3 + \gamma^3 + 1 \leq 0 \end{array} \right\rangle$$

or equivalently

$$\begin{array}{c} (B_i w^1 - \gamma^1 + 1)_+ \cdot (-B_i w^2 + \gamma^2 + 1)_+ = 0 \\ \textbf{or} \\ (-B_i w^1 + \gamma^1 + 1)_+ \cdot (B_i w^3 - \gamma^3 + 1)_+ \cdot \\ (B_i w^4 - \gamma^4 + 1)_+ = 0 \\ \textbf{or} \\ (-B_i w^1 + \gamma^1 + 1)_+ (-B_i w^3 + \gamma^3 + 1)_+ = 0 \end{array} \quad (7)$$

A decision tree exists that strictly classifies all the points in sets $\mathcal{A}$ and $\mathcal{B}$ if and only if the following equation has a feasible solution:

$$\begin{array}{l} \sum\limits_{j=1}^{m} (y_{1_j} + y_{2_j}) \cdot (z_{1_j} + y_{3_j} + z_{4_j}) + \\ \sum\limits_{i=1}^{k} (u_{1_i} + v_{2_i}) \cdot (v_{1_i} + u_{3_i} + u_{4_i}) \cdot (v_{1_i} + v_{3_i}) \;=\; 0 \\ \textit{where} \quad y_{d_j} = (A_j w^d - \gamma^d + 1)_+ \quad j = 1 \ldots m \\ \qquad z_{d_j} = (-A_j w^d + \gamma^d + 1)_+ \\ \qquad u_{d_i} = (B_i w^d - \gamma^d + 1)_+ \quad i = 1 \ldots k \\ \qquad v_{d_i} = (-B_i w^d + \gamma^d + 1)_+ \\ \qquad \textit{for } d = 1 \ldots D \\ \qquad \textit{and } D \;=\; \textit{number of decisions in tree.} \end{array} \quad (8)$$

Furthermore, (w^d, γ^d), $d = 1 \ldots D$, satisfying (8) form the decisions of a tree that strictly classifies all the points in the sets $\mathcal{A}$ and $\mathcal{B}$.

Equivalently, there exists a decision tree with the given structure that correctly classifies the points in sets $\mathcal{A}$ and $\mathcal{B}$ if and only if the following multilinear program has a zero minimum:

$$\begin{array}{ll} \min\limits_{w,\gamma,y,z,u,v} & \frac{1}{m}\sum\limits_{j=1}^{k}(y_{1_j}+y_{2_j})\cdot(z_{1_j}+y_{3_j}+z_{4_j})+ \\ & \frac{1}{k}\sum\limits_{i=1}^{m}(u_{1_i}+v_{2_i})\cdot(v_{1_i}+u_{3_i}+u_{4_i})\cdot(v_{1_i}+v_{3_i}) \\ s.t. & y_{d_j} \geq A_j\, w^d - \gamma^d + 1 \quad j = 1 \ldots m \\ & z_{d_j} \geq -A_j\, w^d + \gamma^d + 1 \\ & u_{d_i} \geq B_i w^d - \gamma^d + 1 \quad i = 1 \ldots k \\ & v_{d_i} \geq -B_i w^d + \gamma^d + 1 \\ & for\ d = 1 \ldots j \\ & y, z, u, v \geq 0 \end{array} \tag{9}$$

The coefficients $\frac{1}{m}$ and $\frac{1}{k}$ were chosen so that (9) is identical to the LP (3) for the single decision case, thus guaranteeing that $w = 0$ is never the unique solution for that case [3]. These coefficients also help to make the method more numerically stable for large training set sizes.

This general approach is applicable to any multivariate binary decision tree used to classify two or more sets. There is an error term for each point in the training set. The error for that point is the product of the errors at each of the leaves. The error at each leaf is the sum of the errors in the decisions along the path to that leaf. If a point is correctly classified at one leaf, the error along the path will be zero, and the product of the leaf errors will be zero. Space does not permit discussion of the general formulation in this paper, thus we refer the reader to [2] for more details.

4 Multilinear Programming

The multilinear program (3) and its more general formulation can be optimized using the iterative linear programming Frank-Wolfe type method proposed in [4]. We outline the method here, and refer the reader to [2] for the mathematical properties of the algorithm.

Consider the problem $\min\limits_{x} f(x)$ subject to $x \in \mathcal{X}$ where $f : R^n \rightarrow R$, $\mathcal{X}$ is a polyhedral set in R^n containing the constraint $x \geq 0$, f has continuous first partial derivatives, and f is bounded below. The Frank-Wolfe algorithm for problem is the following:

Algorithm 4.1 (Frank-Wolfe algorithm [7, 4])
Start with any $x^0 \in \mathcal{X}$. Compute x^{i+1} from x^i as follows.

$$(i) \qquad v^i \in arg\ vertex \min_{x \in \mathcal{X}} \bigtriangledown f(x^i)x$$

$$(ii) \qquad Stop\ if \quad \bigtriangledown f(x^i)v^i = \bigtriangledown f(x^i)x^i$$

$$(iii) \qquad x^{i+1} = (1-\lambda^i)x^i + \lambda^i v^i \ where$$

$$\lambda^i \in arg \min_{0 \leq \lambda \leq 1} f((1-\lambda)x^i + \lambda v^i)$$

In the above algorithm "$arg\ vertex\ \min$" denotes a vertex solution set of the indicated linear program. The algorithm terminates at some x^j that satisfies the 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 [4].

The gradient calculation for the GTO function is straightforward. For example, when Algorithm 4.1 is applied to Problem (9), the following linear subproblem is solved in step (i) with $(\hat{w}, \hat{\gamma}, \hat{y}, \hat{z}, \hat{u}, \hat{v}) = x^i$:

$$\begin{array}{ll} \min\limits_{w,\gamma,y,z,u,v} & \frac{1}{m}\sum\limits_{j=1}^{m}(\hat{y}_{1_j}+\hat{y}_{2_j})(z_{1_j}+y_{3_j}+z_{4_j})+ \\ & \frac{1}{m}\sum\limits_{j=1}^{m}(y_{1_j}+y_{2_j})\cdot(\hat{z}_{1_j}+\hat{y}_{3_j}+\hat{z}_{4_j})+ \\ & \frac{1}{k}\sum\limits_{i=1}^{k}(\hat{u}_{1_i}+\hat{v}_{2_i})\cdot(\hat{v}_{1_i}+\hat{u}_{3_i}+\hat{u}_{4_i}) \\ & \qquad \cdot(v_{1_i}+v_{3_i})+ \\ & \frac{1}{k}\sum\limits_{i=1}^{k}(\hat{u}_{1_i}+\hat{v}_{2_i})\cdot(v_{1_i}+u_{3_i}+u_{4_i})\cdot \\ & \qquad (\hat{v}_{1_i}+\hat{v}_{3_i})+ \\ & \frac{1}{k}\sum\limits_{i=1}^{k}(u_{1_i}+v_{2_i})\cdot(\hat{v}_{1_i}+\hat{u}_{3_i}+\hat{u}_{4_i})\cdot \\ & \qquad (\hat{v}_{1_i}+\hat{v}_{3_i}) \\ s.t. & y_{d_j} \geq A_j w^j - \gamma^d + 1 \quad For\ d = 1, \ldots, D \\ & z_{d_j} \geq -A_j\, w^d + \gamma^d + 1 \quad j = 1 \ldots m \\ & u_{d_i} \geq B_i w^d - \gamma^d + 1 \quad i = 1 \ldots k \\ & v_{d_i} \geq -B_i w^d + \gamma^d + 1 \\ & y, z, u, v \geq 0 \quad fixed\ \hat{y}, \hat{z}, \hat{u}, \hat{v}, \geq 0 \end{array}$$

5 Results and Conclusions

GTO was implemented for general decision trees with fixed structure. In order to test the effectiveness of the optimization algorithm, random problems with known solutions were generated. For a given dimension, a tree with 3 to 7 decision nodes was randomly generated to classify points in the unit cube. Points in the unit cube

were randomly generated and classified and grouped into a training set (500 to 1000 points) and a testing set (5000 points). MSMT, the greedy algorithm discussed in Section 2, was used to generate a greedy tree that correctly classified the training set. The MSMT tree was then pruned to the known structure (i.e. the number of decision nodes) of the tree. The pruned tree was used as a starting point for GTO. The training and testing set error of the MSMT tree, the pruned tree (denoted MSMT-P), and the GTO tree were measured, as was the training time. This experiment was repeated for trees ranging from 3 to 7 nodes in 2 to 25 dimensions. The results were averaged over 10 trials.

We summarize the test results and refer the reader to [2] for more details. Figure 3 presents the average results for randomly generated trees with three decision nodes. These results are typical of those observed in the other experiments. MSMT achieved 100% correctness on the training set but used an excessive number of decisions. The training and testing set accuracy of the pruned trees dropped considerably. The trees once optimized by GTO were significantly better in terms of testing set accuracy than both unpruned and pruned MSMT trees.

The computational results are promising. The Frank-Wolfe algorithm converges in relatively few iterations to an improved solution. However GTO did not always find the global minimum. We expect the problem to have many local minima since it is NP-complete. We plan to investigate using global optimization techniques to avoid local minima. The overall execution time of GTO tends to grow as the problem size increases. Parallel computation can be used to improve the execution time of the expensive LP subproblems. The LP subproblems (e.g. Problem (9)) have a block-separable structure and can be divided into independent LPs solvable in parallel.

We have introduced a non-greedy approach for optimizing decision trees. The GTO algorithm starts with an existing decision tree, fixes the structure of the tree, formulates the error of the tree, and then optimizes that error. An iterative linear programming algorithm performs well on this NP-complete problem. GTO optimizes all the decisions in the tree, and thus has many potential applications such as: decreasing greediness of constructive algorithms, reoptimizing existing trees when additional data is available, pruning greedy decision trees, and incorporating domain knowledge into the decision tree.

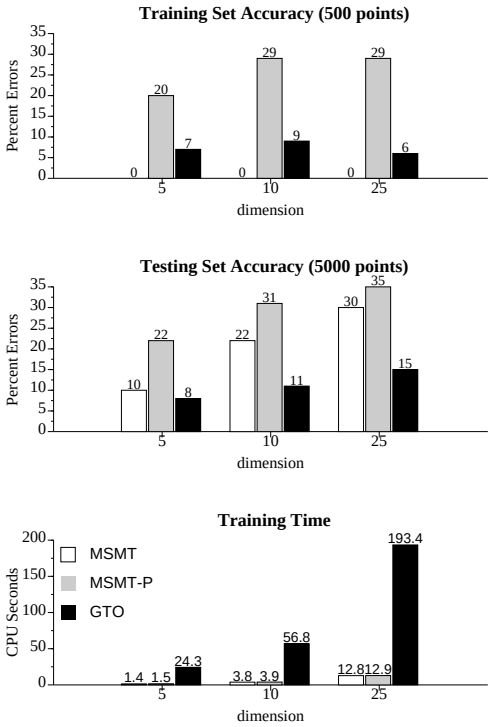


Figure 3: Average results over 10 trials for randomly generated decision trees with 3 decision nodes.

References

[1] 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.

[2] K. P. Bennett. Optimal decision trees through multilinear programming. R.P.I. Math Report No. 214, Rensselaer Polytechnic Institute, Troy, NY, 1994.

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

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

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

[6] C. E. Brodley and P. E. Utgoff. Multivariate decision trees. COINS Technical Report 92-83, University of Massachussets, Amherst, Massachusetts, 1992. To appear in *Machine Learning.*

[7] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

[8] J. H. Friedman. Multivariate adaptive regression splines (with discussion). *Annals of Statistics*, 19:1–141, 1991.

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

[10] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1984.