Optimal classification trees

Citation: Bertsimas, Dimitris, and Jack Dunn. "Optimal Classification Trees." Machine Learning 106.7 (2017): 1039–1082.

As Published: http://dx.doi.org/10.1007/s10994-017-5633-9

Publisher: Springer US

Persistent URL: http://hdl.handle.net/1721.1/110328

Version: Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

Noname manuscript No.
(will be inserted by the editor)

Optimal Classification Trees

Dimitris Bertsimas · Jack Dunn

Received: date / Accepted: date

Abstract State-of-the-art decision tree methods apply heuristics recursively to create each split in isolation, which may not capture well the underlying characteristics of the dataset. The optimal decision tree problem attempts to resolve this by creating the entire decision tree at once to achieve global optimality. In the last 25 years, algorithmic advances in integer optimization coupled with hardware improvements have resulted in an astonishing 800 billion factor speedup in Mixed-Integer Optimization (MIO). Motivated by this speedup, we present *Optimal Classification Trees*, a novel formulation of the decision tree problem using modern MIO techniques that yields the optimal decision tree for axes-aligned splits. We also show the richness of this MIO formulation by adapting it to give *Optimal Classification Trees with Hyperplanes* that generates optimal decision trees with multivariate splits. Synthetic tests demonstrate that these methods recover the true decision tree more closely than heuristics, refuting the notion that optimal methods overfit the training data. We comprehensively benchmark these methods on a sample of 53 datasets from the UCI Machine Learning Repository. We establish that these MIO methods are practically solvable on real-world datasets with sizes in the thousands, and give average absolute improvements in out-of-sample accuracy over CART of 1–2% and 3–5% for the univariate and multivariate cases, respectively. Furthermore, we identify that Optimal Classification Trees are likely to outperform CART by 1.2–1.3% in situations where the CART accuracy is high and we have sufficient training data, while the multivariate version outperforms CART by 4–7% when the CART accuracy or dimension of the dataset is low.

Keywords Classification · Decision trees · Mixed integer optimization

D. Bertsimas
Sloan School of Management and Operations Research Center
Massachusetts Institute of Technology
Cambridge, MA 02139
E-mail: dbertsim@mit.edu

J. Dunn
Operations Research Center
Massachusetts Institute of Technology
Cambridge, MA 02139
E-mail: jackdunn@mit.edu

1 Introduction

Decision trees are one of the most widely-used techniques for classification problems. Guided by the training data $(\mathbf{x}_i, y_i), i = 1, \dots n$, decision trees recursively partition the feature space and assign a label to each resulting partition. The tree is then used to classify future points according to these splits and labels. The key advantage of decision trees over other methods is that they are very interpretable, and in many applications, such as healthcare, this interpretability is often preferred over other methods that may have higher accuracy but are relatively uninterpretable.

The leading work for decision tree methods in classification is CART (Classification and Regression Trees), proposed by Breiman et al (1984), which takes a top-down approach to determining the partitions. Starting from the root node, a split is determined by solving an optimization problem (typically minimizing an impurity measure), before proceeding to recurse on the two resulting child nodes. The main shortcoming of this top-down approach, shared by other popular decision tree methods like C4.5 (Quinlan 1993) and ID3 (Quinlan 1986), is its fundamentally greedy nature. Each split in the tree is determined in isolation without considering the possible impact of future splits in the tree. This can lead to trees that do not capture well the underlying characteristics of the dataset, potentially leading to weak performance when classifying future points.

Another limitation of top-down induction methods is that they typically require pruning to achieve trees that generalize well. Pruning is needed because top-down induction is unable to handle penalties on tree complexity while growing the tree, since powerful splits may be hidden behind weaker splits. This means that the best tree might not be found by the top-down method if the complexity penalty is too high and prevents the first weaker split from being selected. The usual approach to resolve this is to grow the tree as deep as possible before pruning back up the tree using the complexity penalty. This avoids the problem of weaker splits hiding stronger splits, but it means that the training occurs in two phases, growing via a series of greedy decisions, followed by pruning. Lookahead heuristics such as IDX (Norton 1989), LSID3 and ID3-k (Esmeir and Markovitch 2007) also aim to resolve this problem of strong splits being hidden behind weak splits by finding new splits based on optimizing deeper trees rooted at the current leaf, rather than just optimizing a single split. However, it is unclear whether these methods can lead to trees with better generalization ability and avoid the so-called look-ahead pathology of decision tree learning (Murthy and Salzberg 1995a).

These top-down induction methods typically optimize an impurity measure when selecting splits, rather than using the misclassification rate of training points. This seems odd when the misclassification rate is the final objective being targeted by the tree, and indeed is also the measure that is used when pruning the tree. Breiman et al (1984, p.97) explain why impurity measures are used in place of the more natural objective of misclassification:

> ... the [misclassification] criterion does not seem to appropriately reward splits that are more desirable in the context of the continued growth of the tree. ... This problem is largely caused by the fact that our tree growing structure is based on a one-step optimization procedure.

It seems natural to believe that growing the decision tree with respect to the final objective function would lead to better splits, but the use of top-down induction methods and their requirement for pruning prevents the use of this objective.

The natural way to resolve this problem is to form the entire decision tree in a single step, allowing each split to be determined with full knowledge of all other splits. This would result in the *optimal* decision tree for the training data. This is not a new idea; Breiman et al (1984, p.42) noted the potential for such a method:

> Finally, another problem frequently mentioned (by others, not by us) is that the tree procedure is only one-step optimal and not overall optimal. ... If one could search all possible partitions ... the two results might be quite different.
>
> This issue is analogous to the familiar question in linear regression of how well the stepwise procedures do as compared with 'best subsets' procedures. We do not address this problem. At this stage of computer technology, an overall optimal tree growing procedure does not appear feasible for any reasonably sized dataset.

The use of top-down induction and pruning in CART was therefore not due to a belief that such a procedure was inherently better, but instead was guided by practical limitations of the time, given the difficulty of finding an optimal tree. Indeed, it is well-known that the problem of constructing optimal binary decision trees is NP-hard (Hyafil and Rivest 1976). Nevertheless, there have been many efforts previously to develop effective ways of constructing optimal univariate decision trees using a variety of heuristic methods for growing the tree in one step, including linear optimization (Bennett 1992), continuous optimization (Bennett and Blue 1996), dynamic programming (Cox Jr et al 1989; Payne and Meisel 1977), genetic algorithms (Son 1998), and more recently, optimizing an upper bound on the tree error using stochastic gradient descent (Norouzi et al 2015). However, none of these methods have been able to produce certifiably optimal trees in practical times. A different approach is taken by the methods T2 (Auer et al 1995), T3 (Tjortjis and Keane 2002) and T3C (Tzirakis and Tjortjis 2016), a family of efficient enumeration approaches which create optimal non-binary decision trees of depths up to 3. However, trees produced using these enumeration schemes are not as interpretable as binary decision trees, and do not perform significantly better than current heuristic approaches (Tjortjis and Keane 2002; Tzirakis and Tjortjis 2016).

We believe the lack of success in developing an algorithm for optimal decision trees can be attributed to a failure to address correctly the underlying nature of the decision tree problem. Constructing a decision tree involves a series of discrete decisions—whether to split at a node in the tree, which variable to split on—and discrete outcomes—which leaf node a point falls into, whether a point is correctly classified—and as such, the problem of creating an optimal decision tree is best posed as a Mixed-Integer Optimization (MIO) problem.

Continuous optimization methods have been widely used in statistics over the past 40 years, but MIO methods, which have been used to great effect in many other fields, have not had the same impact on statistics. Despite the knowledge that many statistical problems have natural MIO formulations (Arthanari and Dodge 1981), there is the belief within the statistics/machine learning community that MIO problems are intractable even for small to medium instances, which

was true in the early 1970s when the first continuous optimization methods for statistics were being developed.

However, the last twenty-five years have seen an incredible increase in the computational power of MIO solvers, and modern MIO solvers such as GUROBI (Gurobi Optimization Inc. 2015b) and CPLEX (IBM ILOG CPLEX 2014) are able to solve linear MIO problems of considerable size. To quantify this increase, Bixby (2012) tested a set of MIO problems on the same computer using CPLEX 1.2, released in 1991, through CPLEX 11, released in 2007. The total speedup factor was measured to be more than 29,000 between these versions (Bixby 2012; Nemhauser 2013). GUROBI 1.0, an MIO solver first released in 2009, was measured to have similar performance to CPLEX 11. Version-on-version speed comparisons of successive GUROBI releases have shown a speedup factor of nearly 50 between GUROBI 6.5, released in 2015, and GUROBI 1.0 (Bixby 2012; Nemhauser 2013; Gurobi Optimization Inc. 2015a), so the combined machine-independent speedup factor in MIO solvers between 1991 and 2015 is approximately 1,400,000. This impressive speedup factor is due to incorporating both theoretical and practical advances into MIO solvers. Cutting plane theory, disjunctive programming for branching rules, improved heuristic methods, techniques for preprocessing MIOs, using linear optimization as a black box to be called by MIO solvers, and improved linear optimization methods have all contributed greatly to the speed improvements in MIO solvers (Bixby 2012). Coupled with the increase in computer hardware during this same period, a factor of approximately 570,000 (Top500 Supercomputer Sites 2015), the overall speedup factor is approximately 800 billion! This astonishing increase in MIO solver performance has enabled many recent successes when applying modern MIO methods to a selection of these statistical problems (Bertsimas et al 2015; Bertsimas and King 2015a,b; Bertsimas and Mazumder 2014).

The belief that MIO approaches to problems in statistics are not practically relevant was formed in the 1970s and 1980s and it was at the time justified. Given the astonishing speedup of MIO solvers and computer hardware in the last twenty-five years, the mindset of MIO as theoretically elegant but practically irrelevant is no longer supported. In this paper, we extend this re-examination of statistics under a modern optimization lens by using MIO to formulate and solve the decision tree training problem, and provide empirical evidence of the success of this approach.

One key advantage of using MIO is the richness offered by the modeling framework. For example, consider *multivariate* (or *oblique*) decision trees, which split on multiple variables at a time, rather than the *univariate* or *axis-aligned* trees that are more common. These multivariate splits are often much stronger than univariate splits, however the problem of determining the split at each node is much more complicated than the univariate case, as a simple enumeration is no longer possible. Many approaches to solving this problem have been proposed, including using logistic regression (Truong 2009), support vector machines (Bennett and Blue 1998), simulated annealing (Heath et al 1993), linear discriminant analysis (Loh and Shih 1997; López-Chau et al 2013), Householder transformations (Wickramarachchi et al 2016), and perturbation of existing univariate trees (Breiman et al 1984; Murthy et al 1994). Most of these approaches do not have easily accessible implementations that can be used on practically-sized datasets and as such, the use of multivariate decision trees in the statistics/machine learning community has been limited. We also note that these multivariate decision tree approaches

share the same major flaw as univariate decision trees, in that the splits are formed one-by-one using top-down induction, and so the choice of split cannot be guided by the possible influence of future splits. However, when viewed from an MIO perspective the problems of generating univariate and multivariate problems are very similar, and the second is in fact simpler than the first, allowing a practical way to generate such trees which has not previously been possible for any heuristic. The flexibility of modeling the problem using MIO allows many such extensions to incorporated very easily.

Our goal in this paper is to demonstrate that formulating and solving the decision tree problem using MIO is a tractable approach and leads to practical solutions that outperform the classical approaches, often significantly.

We summarize our contributions in this paper below:

1. We present a new, novel formulation of the classical univariate decision tree problem as an MIO problem that motivates our new classification method, *Optimal Classification Trees* (OCT). We also show that by relaxing constraints in this model we obtain the multivariate decision tree problem, and that the resulting classification method, *Optimal Classification Trees with Hyperplanes* (OCT-H), is easier to train than its univariate counterpart, a fact that is not true about heuristic approaches for generating multivariate decision trees.
2. Using a range of tests with synthetic data comparing OCT against CART, we demonstrate that solving the decision tree problem to optimality yields trees that better reflect the ground truth in the data, refuting the belief that such optimal methods will simply overfit to the training set and not generalize well.
3. We demonstrate that our MIO methods outperform classical decision tree methods in practical applications. We comprehensively benchmark both OCT and OCT-H against the state-of-the-art CART on a sample of 53 datasets from the UCI Machine Learning Repository. We show that across this sample, the OCT and OCT-H problems are practically solvable for datasets with sizes in the thousands and yield higher out-of-sample accuracy than CART, with average absolute improvements over CART of 1–2% and 3–5% for OCT and OCT-H, respectively, across all datasets depending on the depth of tree used.
4. We provide a comparison of OCT and OCT-H to Random Forests. Across all 53 datasets, OCT closes the gap between CART and Random Forests by about one-sixth, and OCT-H by about half. Furthermore, for the 31 datasets with two or three classes, and $p \leq 25$, OCT-H and Random Forests are comparable in accuracy.
5. To provide guidance to machine learning practitioners, we present simple decision rules using characteristics of the dataset that predict with high accuracy when the Optimal Tree methods will deliver consistent and significant accuracy improvements over CART. OCT is highly likely to improve upon CART by 2–4% when the CART accuracy is low, and by 1.2–1.3% when the CART accuracy is high and sufficient training data is availble. OCT-H improves upon CART by 4–7% when the CART accuracy is low or the dimension of the dataset is small.

We note that there have been efforts in the past to apply MIO methods to classification problems. CRIO (Classification and Regression via Integer Optimization), proposed by Bertsimas and Shioda (2007), uses MIO to partition and classify the data points. CRIO was not able to solve the classification problems to

provable optimality for moderately-sized classification problems, and the practical improvement over CART was not significant. In contrast, in this paper we present a very different MIO approach based around solving the same problem that CART seeks to solve, and this approach provides material improvement over CART for a variety of datasets.

The structure of the paper is as follows. In Section 2, we review decision tree methods and formulate the problem of optimal tree creation within an MIO framework. In Section 3, we present a complete training algorithm for optimal tree classification methods. In Section 4, we extend the MIO formulation to consider trees with multivariate splits. In Section 5, we conduct a range of experiments using synthetic datasets to evaluate the performance of our method in recovering a known ground truth. In Section 6, we perform a variety of computational tests on real-world datasets to benchmark our performance against classical decision tree methods in practical applications. In Section 7, we include our concluding remarks.

2 Optimal Decision Tree Formulation using MIO

In this section, we first give an overview of the classification problem and the approach taken by typical decision tree methods. We then present the problem that CART seeks to solve as a formal optimization problem and develop an MIO model formulation for this problem allowing us to solve it to optimality, providing the basis for our method *Optimal Classification Trees* (OCT).

2.1 Overview of Classification Problem and Decision Trees

We are given the training data $(\mathbf{X}, \mathbf{Y})$, containing n observations $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, n$, each with p features $\mathbf{x}_i \in \mathbb{R}^p$ and a class label $y_i \in \{1, \ldots, K\}$ indicating which of K possible labels is assigned to this point. We assume without loss of generality that the values for each dimension across the training data are normalized to the 0–1 interval, meaning each $\mathbf{x}_i \in [0, 1]^p$.

Decision tree methods seek to recursively partition $[0, 1]^p$ to yield a number of hierarchical, disjoint regions that represent a classification tree. An example of a decision tree is shown in Figure 1. The final tree is comprised of branch nodes and leaf nodes:

- Branch nodes apply a split with parameters $\mathbf{a}$ and b. For a given point i, if $\mathbf{a}^T\mathbf{x}_i < b$ the point will follow the left branch from the node, otherwise it takes the right branch. A subset of methods, including CART, produce *univariate* or *axis-aligned* decision trees which restrict the split to a single dimension, i.e., a single component of $\mathbf{a}$ will be 1 and all others will be 0.
- Leaf nodes are assigned a class that will determine the prediction for all data points that fall into the leaf node. The assigned class is almost always taken to be the class that occurs most often among points contained in the leaf node.

Classical decision tree methods like CART, ID3, and C4.5 take a top-down approach to building the tree. At each step of the partitioning process, they seek to find a split that will partition the current region in such a way to maximize

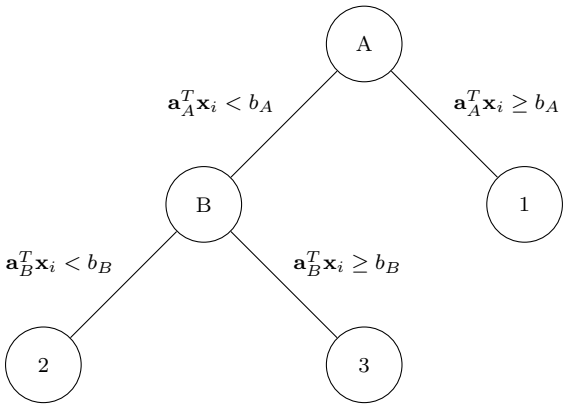


Fig. 1 An example of a decision tree with two partition nodes and three leaf nodes

a so-called splitting criterion. This criterion is often based on the label impurity of the data points contained in the resulting regions instead of minimizing the resulting misclassification error, which as discussed earlier is a byproduct of using a top-down induction process to grow the tree. The algorithm proceeds to recursively partition the two new regions that are created by the hyperplane split. The partitioning terminates once any one of a number of stopping criteria are met. The criteria for CART are as follows:

- It is not possible to create a split where each side of the partition has at least a certain number of nodes, $N_{\min}$;
- All points in the candidate node share the same class.

Once the splitting process is complete, a class label $1, \ldots, K$ is assigned to each region. This class will be used to predict the class of any points contained inside the region. As mentioned earlier, this assigned class will typically be the most common class among the points in the region.

The final step in the process is pruning the tree in an attempt to avoid overfitting. The pruning process works upwards through the partition nodes from the bottom of the tree. The decision of whether to prune a node is controlled by the so-called *complexity parameter*, denoted by α, which balances the additional complexity of adding the split at the node against the increase in predictive accuracy that it offers. A higher complexity parameter leads to more and more nodes being pruned off, resulting in smaller trees.

As discussed previously, this two-stage growing and pruning procedure for creating the tree is required when using a top-down induction approach, as otherwise the penalties on tree complexity may prevent the method from selecting a weaker split that then allows a selection of a stronger split in the next stage. In this sense, the strong split is hidden behind the weak split, and this strong split may be passed over if the growing-then-pruning approach is not used.

Using the details of the CART procedure, we can state the problem that CART attempts to solve as a formal optimization problem. There are two parameters in this problem. The tradeoff between accuracy and complexity of the tree is

controlled by the complexity parameter α, and the minimum number of points we require in any leaf node is given by $N_{\min}$. Given these parameters and the training data $(\mathbf{x}_i, y_i), i = 1, \ldots, n$, we seek a tree T that solves the problem:

$$\begin{aligned} \min \quad & R_{xy}(T) + \alpha|T| \\ \text{s.t.} \quad & N_x(l) \geq N_{\min} \qquad \forall l \in \text{leaves}(T) \end{aligned} \tag{1}$$

where $R_{xy}(T)$ is the misclassification error of the tree T on the training data, $|T|$ is the number of branch nodes in tree T, and $N_x(l)$ is the number of training points contained in leaf node l. We refer to this problem as the *optimal tree problem*.

Notice that if we can solve this problem in a single step we obviate the need to use an impurity measure when growing the tree, and also remove the need to prune the tree after creation, as we have already accounted for the complexity penalty while growing the tree.

We briefly note that our choice to use CART to define the optimal tree problem was arbitrary, and one could similarly define this problem based on another method like C4.5; we simply use this problem to demonstrate the advantages of taking a problem that is traditionally solved by a heuristic and instead solving it to optimality. Additionally, we note that the experiments of Murthy and Salzberg (1995b) found that CART and C4.5 did not differ significantly in any measure of tree quality, including out-of-sample accuracy, and so we do not believe our choice of CART over C4.5 to be an important one.

2.2 Formulating Optimal Tree Creation as an MIO Problem

As mentioned previously, the top-down, greedy nature of state-of-the-art decision tree creation algorithms can lead to solutions that are only locally optimal. In this section, we first argue that the natural way to pose the task of creating the globally optimal decision tree is as an MIO problem, and then proceed to develop such a formulation.

To see that the most natural representation for formulating the optimal decision tree problem (1) is using MIO, we note that at every step in tree creation, we are required to make a number of discrete decisions:

- At every new node, we must choose to either branch or stop.
- After choosing to stop branching at a node, we must choose a label to assign to this new leaf node.
- After choosing to branch, we must choose which of the variables to branch on.
- When classifying the training points according to the tree under construction, we must choose to which leaf node a point will be assigned such that the structure of the tree is respected.

Formulating this problem using MIO allows us to model all of these discrete decisions in a single problem, as opposed to recursive, top-down methods that must consider these decision events in isolation. Modeling the construction process in this way allows us to consider the full impact of the decisions being made at the top of the tree, rather than simply making a series of locally optimal decisions, also avoiding the need for pruning and impurity measures.

We will next formulate the optimal tree creation problem (1) as an MIO problem. Consider the problem of trying to construct an optimal decision tree with a

maximum depth of D. Given this depth, we can construct the *maximal tree* of this depth, which has $T = 2^{(D+1)} - 1$ nodes, which we index by $t = 1, \ldots, T$. Figure 2 shows the maximal tree of depth 2.

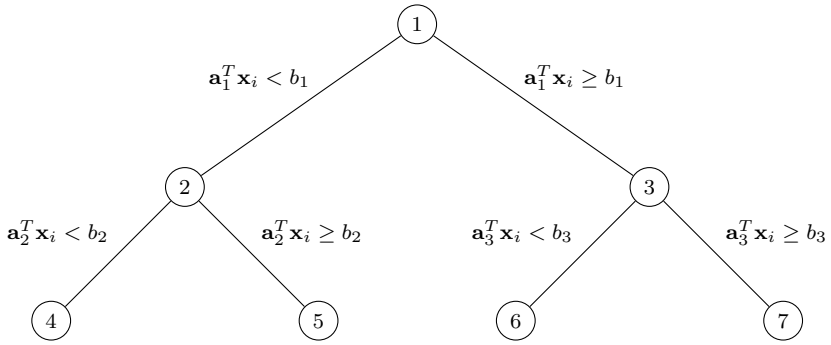


Fig. 2 The maximal tree for depth $D = 2$

We use the notation $p(t)$ to refer to the parent node of node t, and $A(t)$ to denote the set of ancestors of node t. We also define $A_L(t)$ as the set of ancestors of t whose left branch has been followed on the path from the root node to t, and similarly $A_R(t)$ is the set of right-branch ancestors, such that $A(t) = A_L(t) \cup A_R(t)$. For example, in the tree in Figure 2, $A_L(5) = \{1\}$, $A_R(5) = \{2\}$, and $A(5) = \{1, 2\}$.

We divide the nodes in the tree into two sets:

Branch nodes: Nodes $t \in \mathcal{T}_B = \{1, \ldots, \lfloor T/2 \rfloor\}$ apply a split of the form $\mathbf{a}^\mathsf{T}\mathbf{x} < b$. Points that satisfy this split follow the left branch in the tree, and those that do not follow the right branch.

Leaf nodes: Nodes $t \in \mathcal{T}_L = \{\lfloor T/2 \rfloor + 1, \ldots, T\}$ make a class prediction for each point that falls into the leaf node.

We track the split applied at node $t \in \mathcal{T}_B$ with variables $\mathbf{a}_t \in \mathbb{R}^p$ and $b_t \in \mathbb{R}$. We will choose to restrict our model to univariate decision trees (like CART), and so the hyperplane split at each node should only involve a single variable. This is enforced by setting the elements of $\mathbf{a}_t$ to be binary variables that sum to 1. We want to allow the option of not splitting at a branch node. We use the indicator variables $d_t = \mathbb{1}\{\text{node } t \text{ applies a split}\}$ to track which branch nodes apply splits. If a branch node does not apply a split, then we model this by setting $\mathbf{a}_t = \mathbf{0}$ and $b_t = 0$. This has the effect of forcing all points to follow the right split at this node, since the condition for the left split is $0 < 0$ which is never satisfied. This allows us to stop growing the tree early without introducing new variables to account for points ending up at the branch node—instead we send them all the same direction down the tree to end up in the same leaf node. We enforce this with the following constraints:

$$\sum_{j=1}^{p} a_{jt} = d_t, \quad \forall t \in \mathcal{T}_B, \tag{2}$$

$$0 \leq b_t \leq d_t, \quad \forall t \in \mathcal{T}_B, \tag{3}$$

$$a_{jt} \in \{0,1\}, \quad j = 1,\ldots,p, \ \forall t \in \mathcal{T}_B, \tag{4}$$

where the second inequality is valid for b_t since we have assumed that each $\mathbf{x}_i \in [0,1]^p$, and we know that $\mathbf{a}_t$ has one element that is 1 if and only if $d_t = 1$, with the remainder being 0. Therefore we it is always true that $0 \leq \mathbf{a}_t^\mathsf{T}\mathbf{x}_i \leq d_t$ for any i and t, and we need only consider values for b_t in this same range.

Next, we will enforce the hierarchical structure of the tree. We restrict a branch node from applying a split if its parent does not also apply a split.

$$d_t \leq d_{p(t)}, \quad \forall t \in \mathcal{T}_B \setminus \{1\}, \tag{5}$$

where no such constraint is required for the root node.

We have constructed the variables that allow us to model the tree structure using MIO; we now need to track the allocation of points to leaves and the associated errors that are induced by this structure.

We introduce the indicator variables $z_{it} = \mathbb{1}\{\mathbf{x}_i \text{ is in node } t\}$ to track the points assigned to each leaf node, and we will then use the indicator variables $l_t = \mathbb{1}\{\text{leaf } t \text{ contains any points}\}$ to enforce a minimum number of points at each leaf, given by $N_{\min}$:

$$z_{it} \leq l_t, \qquad t \in \mathcal{T}_B, \tag{6}$$

$$\sum_{i=1}^{n} z_{it} \geq N_{\min} l_t, \quad t \in \mathcal{T}_B. \tag{7}$$

We also force each point to be assigned to exactly one leaf:

$$\sum_{t \in \mathcal{T}_L} z_{it} = 1, \quad i = 1,\ldots,n. \tag{8}$$

Finally, we apply constraints enforcing the splits that are required by the structure of the tree when assigning points to leaves:

$$\mathbf{a}_m^\mathsf{T}\mathbf{x}_i < b_t + M_1(1 - z_{it}), \quad i = 1,\ldots,n, \ \forall t \in \mathcal{T}_B, \ \forall m \in A_L(t), \tag{9}$$

$$\mathbf{a}_m^\mathsf{T}\mathbf{x}_i \geq b_t - M_2(1 - z_{it}), \quad i = 1,\ldots,n, \ \forall t \in \mathcal{T}_B, \ \forall m \in A_R(t), \tag{10}$$

Note that the constraints (9) use a strict inequality which is not supported by MIO solvers, so this must be converted into a form that does not use a strict inequality. To do this we can add a small constant ϵ to the left-hand-side of (9) and change the inequality to be non-strict:

$$\mathbf{a}_m^T\mathbf{x}_i + \epsilon \leq b_m + M_1\left(1 - z_{it}\right), \quad i = 1,\ldots,n, \ \forall t \in \mathcal{T}_B, \ \forall m \in A_L(t) \tag{11}$$

However, if ϵ is too small, this could cause numerical instabilities in the MIO solver, so we seek to make ϵ as big as possible without affecting the feasibility of any valid solution to the problem. We can achieve this by specifying a different ϵ_j for each feature j. The largest valid value is the smallest non-zero distance between adjacent values of this feature. To find this, we sort the values of the jth feature and take

$$\epsilon_j = \min\left\{ x_j^{(i+1)} - x_j^{(i)} \;\middle|\; x_j^{(i+1)} \neq x_j^{(i)}, i = 1,\ldots,n-1 \right\},$$

where $x_j^{(i)}$ is the ith largest value in the jth feature. We can then use these values for ϵ in the constraint, where the value of ϵ_j that is used is selected according to the feature we are using for this split:

$$\mathbf{a}_m^T(\mathbf{x}_i + \boldsymbol{\epsilon}) \leq b_m + M_1\,(1 - z_{it})\,, \quad i = 1, \ldots, n,\ \forall t \in \mathcal{T}_B,\ \forall m \in A_L(t) \tag{12}$$

We must also specify values for the big-M constants M_1 and M_2. As mentioned previously, we know that both $\mathbf{a}_t^T\mathbf{x}_i \in [0,1]$ and $b_t \in [0,1]$, and so the largest possible value of $\mathbf{a}_t^T(\mathbf{x}_i + \epsilon) - b_t$ is $1 + \epsilon_{\max}$, where $\epsilon_{\max} = \max_j\{\epsilon_j\}$. We can therefore set $M_1 = 1 + \epsilon_{\max}$. Similarly, we have the largest possible value of $b_t - \mathbf{a}_t^T x_i$ is 1, and so we can set $M_2 = 1$. This gives the following final constraints that will enforce the splits in the tree:

$$\mathbf{a}_m^T(\mathbf{x}_i + \boldsymbol{\epsilon}) \leq b_m + (1 + \epsilon_{\max})\,(1 - z_{it})\,, \quad i = 1, \ldots, n,\ \forall t \in \mathcal{T}_L,\ \forall m \in A_L(t), \tag{13}$$

$$\mathbf{a}_m^{\mathsf{T}}\mathbf{x}_i \geq b_m - (1 - z_{it}), \qquad i = 1, \ldots, n,\ \forall t \in \mathcal{T}_L,\ \forall m \in A_R(t). \tag{14}$$

The objective is to minimize the misclassification error, so an incorrect label prediction has cost 1, and a correct label prediction has cost 0. Let us define the matrix $\mathbf{Y}$ using the data, where

$$Y_{ik} = \begin{cases} +1, & \text{if } y_i = k \\ -1, & \text{otherwise} \end{cases}, \quad k = 1, \ldots, K, \quad i = 1, \ldots, n.$$

We set N_{kt} to be the number of points of label k in node t, and N_t to be the total number of points in node t:

$$N_{kt} = \frac{1}{2}\sum_{i=1}^{n}(1 + Y_{ik})z_{it}, \quad k = 1, \ldots, K,\ t \in \mathcal{T}_L, \tag{15}$$

$$N_t = \sum_{i=1}^{n} z_{it}, \qquad \forall t \in \mathcal{T}_L. \tag{16}$$

We need to assign a label to each leaf node t in the tree, which we denote with $c_t \in \{1, \ldots, K\}$. It is clear that the optimal label to predict is the most common of the labels among all points assigned to the node:

$$c_t = \underset{k=1,\ldots,K}{\arg\max}\{N_{kt}\} \tag{17}$$

We will use binary variables c_{kt} to track the prediction of each node, where $c_{kt} = \mathbb{1}\{c_t = k\}$. We must make a single class prediction at each leaf node that contains points:

$$\sum_{k=1}^{K} c_{kt} = l_t, \quad \forall t \in \mathcal{T}_L. \tag{18}$$

Since we know how to make the optimal prediction at each leaf t using (17), the optimal misclassification loss in each node, denoted L_t is going to be equal to the number of points in the node less the number of points of the most common label:

$$L_t = N_t - \max_{k=1,\ldots,K}\{N_{kt}\} = \min_{k=1,\ldots,K}\{N_t - N_{kt}\}, \tag{19}$$

which can be linearized to give

$$L_t \geq N_t - N_{kt} - M(1 - c_{kt}), \quad k = 1, \ldots, K, \ \forall t \in \mathcal{T}_L, \tag{20}$$
$$L_t \leq N_t - N_{kt} + Mc_{kt}, \quad k = 1, \ldots, K, \ \forall t \in \mathcal{T}_L, \tag{21}$$
$$L_t \geq 0, \quad \forall t \in \mathcal{T}_L, \tag{22}$$

where again M is a sufficiently large constant that makes the constraint inactive depending on the value of c_{kt}. Here, we can take $M = n$ as a valid value.

The total misclassification cost is therefore $\sum_{t \in \mathcal{T}_L} L_t$, and the complexity of the tree is the number of splits included in the tree, given by $\sum_{t \in \mathcal{T}_B} d_t$. Following CART, we normalize the misclassification against the baseline accuracy, $\hat{L}$, obtained by simply predicting the most popular class for the entire dataset. This makes the effect of α independent of the dataset size. This means the objective from problem (1) can be written:

$$\min \quad \frac{1}{\hat{L}} \sum_{t \in \mathcal{T}_L} L_t + \alpha \sum_{t \in \mathcal{T}_B} d_t. \tag{23}$$

Putting all of this together gives the following MIO formulation for problem (1), which we call the *Optimal Classification Trees* (OCT) model:

$$\min \quad \frac{1}{\hat{L}} \sum_{t \in \mathcal{T}_L} L_t + \alpha \sum_{t \in \mathcal{T}_B} d_t \tag{24}$$
$$\text{s.t.} \quad L_t \geq N_t - N_{kt} - n(1 - c_{kt}), \quad k = 1, \ldots, K, \ \forall t \in \mathcal{T}_L,$$
$$L_t \leq N_t - N_{kt} + nc_{kt}, \quad k = 1, \ldots, K, \ \forall t \in \mathcal{T}_L,$$
$$L_t \geq 0, \quad \forall t \in \mathcal{T}_L,$$
$$N_{kt} = \frac{1}{2} \sum_{i=1}^{n} (1 + Y_{ik}) z_{it}, \quad k = 1, \ldots, K, \ \forall t \in \mathcal{T}_L,$$
$$N_t = \sum_{i=1}^{n} z_{it}, \quad \forall t \in \mathcal{T}_L,$$
$$\sum_{k=1}^{K} c_{kt} = l_t, \quad \forall t \in \mathcal{T}_L,$$
$$\mathbf{a}_m^\top \mathbf{x}_i \geq b_t - (1 - z_{it}), \quad i = 1, \ldots, n, \ \forall t \in \mathcal{T}_B, \ \forall m \in A_R(t),$$
$$\mathbf{a}_m^\top (\mathbf{x}_i + \boldsymbol{\epsilon}) \leq b_t + (1 + \epsilon_{\max})(1 - z_{it}), \quad i = 1, \ldots, n, \ \forall t \in \mathcal{T}_B, \ \forall m \in A_L(t),$$
$$\sum_{t \in \mathcal{T}_L} z_{it} = 1, \quad i = 1, \ldots, n,$$
$$z_{it} \leq l_t, \quad \forall t \in \mathcal{T}_L,$$
$$\sum_{i=1}^{n} z_{it} \geq N_{\min} l_t, \quad \forall t \in \mathcal{T}_L,$$
$$\sum_{j=1}^{p} a_{jt} = d_t, \quad \forall t \in \mathcal{T}_B,$$
$$0 \leq b_t \leq d_t, \quad \forall t \in \mathcal{T}_B,$$

$$
\begin{aligned}
&d_t \leq d_{p(t)}, && \forall t \in \mathcal{T}_B \setminus \{1\},\\
&z_{it}, l_t \in \{0,1\}, && i = 1,\ldots,n,\ \forall t \in \mathcal{T}_L,\\
&a_{jt}, d_t \in \{0,1\}, && j = 1,\ldots,p,\ \forall t \in \mathcal{T}_B.
\end{aligned}
$$

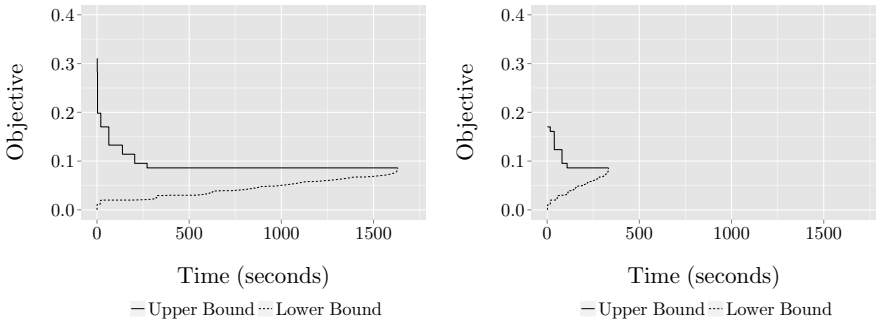
This model as presented is in a form that can be directly solved by any MIO solver. The difficulty of the model is primarily determined by the number of binary variables z_{it}, which is $n \cdot 2^D$. Empirically we observe that we can find high-quality solutions in minutes for depths up to 4 for datasets with thousands of points. Beyond this depth or dataset size, the rate of finding solutions is slower, and more time is required.

There are three hyper-parameters that need to be specified: the maximum depth D, the minimum leaf size $N_{\min}$, and the complexity parameter α. We will present in Section 3 effective methods for tuning these parameters in validation.

2.3 Improving MIO Performance using Warm Starts

MIO solvers benefit greatly when supplied an integer-feasible solution as a *warm start* for the solution process. Injecting a strong warm start solution before starting the solver greatly increases the speed with which the solver is able to generate strong feasible solutions (Bertsimas and Weismantel 2005). The warm start provides a strong initial upper bound on the optimal solution that allows more of the search tree to be pruned, and it also provides a starting point for local search heuristics. The benefit realized increases with the quality of the warm start, so it is desirable to be able to quickly and heuristically find a strong integer-feasible solution before solving.

To demonstrate the effectiveness of warm starts, Figure 3 shows an example of the typical evolution of upper and lower bounds as we solve the MIO problem (24) for the optimal tree of depth 2 on the "Wine" dataset, which has $n = 178$ and $p = 13$. We see when no warm start is supplied, the upper bound decreases gradually until the eventual optimal solution is found after 270 seconds. An additional $1,360$ seconds are required to prove the optimality of this solution. When we inject a heuristic solution (in this case, the CART solution) as a warm start, the same optimal solution is found after just 105 seconds, and it takes an additional 230 seconds to prove optimality. The total time required to find and prove optimality in this example decreases by a factor of 5 when the warm start is added, and the time taken to find the optimal solution decreases by a factor of around 2.5, showing that using high-quality warm start solutions can have a significant effect on the MIO solution times. Additionally, we see that the optimal tree solution has an objective with roughly half the error of the CART warm start, showing that the CART solutions can indeed be far from optimality in-sample. Finally, we observe that the majority of the time is spent in proving that the solution is optimal. A proof of optimality is good to have for comparison to other methods, but is not necessarily required in practice when evaluating a classifier (note that other methods make no claims as to their optimality). It is therefore a reasonable option to terminate the solve early once the best solution has remained unchanged for some time, because this typically indicates the solution is indeed optimal or close to it, and proving optimality may take far more time.



(a) Without warm start **(b)** With warm start

Fig. 3 Comparison of upper and lower bound evolution while solving MIO problem (24) with and without warm starts for a tree of depth $D = 2$ for the Wine dataset with $n = 178$ and $p = 13$

We already have access to good heuristic methods for the MIO problem (24); we can use CART to generate these warm start solutions. Given a solution from CART, it is simple to construct a corresponding feasible solution to the MIO problem (24) using the splits from CART to infer the values of the remaining variables in the MIO problem.

By design, the parameters $N_{\min}$ and α have the same meaning in CART and in problem (24), so we can run CART with these parameters to generate a good solution. We must then prune the splits on the CART solution until it is below the maximum depth D for the MIO problem to ensure it is feasible.

We can also use MIO solutions that we have previously generated as warm starts. In particular, if we have a solution generated for a depth D, this solution is a valid warm start for depth $D + 1$. This is important because the problem difficulty increases with the depth, so for larger depths it may be advantageous to run the MIO problem with a smaller depth to generate a strong warm start. This technique is used heavily in the validation procedure described in Section 3.

3 Using Optimal Classification Trees for Classification

In this section, we present the implementation details of a full algorithm for training a decision tree classifier using the Optimal Classification Trees MIO problem from Section 2. In particular, we devise a procedure for effectively tuning the values of the hyperparameters of this model.

The most important parameter to choose is the maximum depth of the tree D. As discussed in Section 2, the depth has a very large effect on the difficulty of the resulting problem, and it is advantageous to reuse solutions from lower depths as warm starts for the higher-depth problems. To take advantage of this, we will specify a maximum depth $D_{\max}$, and start building trees from depth 2 up to this maximum depth, maintaining a pool of possible warm start solutions.

We also have to tune the value of the complexity parameter α. Recall that the objective is given by

$$\min \quad \frac{1}{\hat{L}} \sum_{t \in \mathcal{T}_L} L_t + \alpha \sum_{t \in \mathcal{T}_B} d_t. \tag{25}$$

It is hard to tune the value of the continuous-valued parameter α. A typical approach would be to discretize the search space and test each value. In our case, this would involve solving the MIO problem for each value, which is expensive. Additionally, multiple values of α might give the same solution which would mean that solving these problems was unnecessary. Ideally we could simply search over the "critical" values of α that would lead to different solutions, minimizing the number of MIO problems that need to be solved.

Observe that the penalty term can be rewritten into constraint form as follows:

$$\begin{aligned} \min \quad & \frac{1}{\hat{L}} \sum_{t \in \mathcal{T}_L} L_t \\ \text{s.t.} \quad & \sum_{t \in \mathcal{T}_B} d_t \leq C, \end{aligned}$$

where C is some constant corresponding to the value of α.

Note that the term in this constraint is the number of splits in the model and is integer-valued. This means we need only test values of C that are integral. We can therefore search over $C = 1, \ldots, C_{\max}$ to generate the entire range of possible solutions, where $C_{\max} = 2^D - 1$ is the maximum number of possible splits in the solution. Additionally, a solution to the problem for C is a feasible warm start solution for the problem at $C + 1$, so the solutions generated during this search can be stored and reused during the remaining search.

It can happen that the solution for a particular value of C is not actually an optimal solution to the original problem (24) for any value of α. This means it is dominated by solutions for other values of C. Such solutions can be removed in a post-processing step to ensure that only solutions optimal for the original problem for any value of α remain.

Given this set of optimal solutions for each value of C, we evaluate each on a validation set and identify the solution that performs best. We can then calculate the interval for α in which this solution is the minimizer for problem (24). We use the midpoint of this interval as the final tuned value for α. In case of a tie, we take the union of the intervals for all such solutions and take the midpoint of this union interval.

An overview of the complete process for tuning the parameters using a validation set follows:

1. Set the maximal depth for the MIO problem, $D_{\max}$, and the minimum leaf size $N_{\min}$.
2. For $D = 1, \ldots, D_{\max}$:
   (a) For $C = 1, \ldots, 2^D - 1$:
      i. Run CART using $N_{\min}$ with $\alpha = 0$. Trim the solution to depth D and to a maximum of C splits.
      ii. Search through pool of candidate warm starts (including CART) and choose the one with the lowest error.

iii. Solve MIO problem for depth D and C splits using the selected warm start.
iv. Add the MIO solution to the warm start pool.

3. Post-process the solution pool to remove all solutions that are not optimal to (24) for any value of α.
4. Identify the best performing solution on a validation set, and the range of α for which this solution is optimal. Use the midpoint of this interval as the tuned value of α.

We have not addressed how to tune the value of $N_{\min}$ as this was not required for the computational experiments in Section 6. In theory it would be simple to use a similar approach for reusing solutions as warm starts by noting that a solution for a value of $N_{\min}$ is also valid for any smaller value of this parameter. In this way, it would be best to search over the $N_{\min}$ values in reverse order, and this could be carried out as the outermost loop in the search procedure.

Having tuned the parameter values using the training and validation sets, we proceed to use these parameters to train a final tree on the combined training and validation sets. Following a similar procedure as in the validation, we use solutions from lower depths to warm start higher depths. For each depth $D = 1, \ldots, D_{\max}$, we run the MIO problem, with the warm start being either CART trimmed to this depth, or the best MIO solution from a lower depth. Finally, we take the final solution and evaluate this on the test set to get the out-of-sample error.

4 Optimal Multivariate Decision Trees using MIO

So far, we have only considered decision trees that use a single variable in their splits at each node, known as *univariate decision trees*. In this section, we show that it is simple to extend our MIO formulation for univariate trees to yield a problem for determining the optimal multivariate decision tree.

4.1 Formulating the Multivariate Optimal Tree Problem

In a multivariate decision tree, we are no longer restricted to choosing a single variable upon which to split, and instead can choose a general hyperplane split at each node. The variables $\mathbf{a}_t$ will be used to model the split at each node as before, except we relax (4) and instead choose $\mathbf{a}_t \in [-1, 1]^p$ at each branch node t. We must modify (2) to account for the the possibility these elements are negative by dealing with the absolute values instead:

$$\sum_{j=1}^{p} |a_{jt}| \leq d_t, \quad \forall t \in \mathcal{T}_B,$$

which can be linearized using auxiliary variables to track the value of $|a_{jt}|$:

$$\begin{aligned} &\sum_{j=1}^{p} \hat{a}_{jt} \leq d_t, && \forall t \in \mathcal{T}_B, \\ &\hat{a}_{jt} \geq a_{jt}, && j = 1, \ldots, p, \ \forall t \in \mathcal{T}_B, \end{aligned}$$

$$\hat{a}_{jt} \geq -a_{jt}, \qquad j = 1, \ldots, p, \ \forall t \in \mathcal{T}_B.$$

As before, these constraints force the split to be all zeros if $d_t = 0$ and no split is applied.

We now have that $\mathbf{a}_t^T \mathbf{x}_i \in [-1, 1]$, so we replace (3) with:

$$-d_t \leq b_t \leq d_t, \quad \forall t \in \mathcal{T}_B.$$

Now we consider the split constraints (9) and (14). Previously we had that the range of $(\mathbf{a}_t^T \mathbf{x}_i - b_t)$ was $[-1, 1]$, whereas it is now $[-2, 2]$. This means we need $M = 2$ to ensure that the constraint is trivially satisfied when $z_{it} = 0$. The constraints therefore become:

$$\mathbf{a}_m^T \mathbf{x}_i < b_m + 2\,(1 - z_{it}), \quad i = 1, \ldots, n, \ \forall t \in \mathcal{T}_B, \ \forall m \in A_L(t), \tag{26}$$

$$\mathbf{a}_m^T \mathbf{x}_i \geq b_m - 2\,(1 - z_{it}), \quad i = 1, \ldots, n, \ \forall t \in \mathcal{T}_B, \ \forall m \in A_R(t), \tag{27}$$

Finally, as before we need to convert the strict inequality in (26) to a non-strict version. We do this by introducing a sufficiently small constant μ:

$$\mathbf{a}_m^T \mathbf{x}_i + \mu \leq b_m + (2 + \mu)\,(1 - z_{it}), \quad i = 1, \ldots, n, \ \forall t \in \mathcal{T}_B, \ \forall m \in A_L(t),$$

Note that we need to include μ in the rightmost term to ensure the constraint is always satisfied when $z_{ik} = 0$. Additionally, unlike in the univariate case, we cannot choose a value for μ in an intelligent manner, and instead need to choose a small constant. Choosing μ too small can lead to numerical issues in the MIO solver, while too large reduces the size of the feasible region, potentially reducing the quality of the optimal solution. We take $\mu = 0.005$ as a compromise between these extremes.

Previously, we penalized the number of splits in the tree. In the multivariate regime where a single split may use multiple variables, it seems sensible to generalize this to instead penalize the total number of variables used in the splits. To achieve this, we introduce binary variables s_{jt} to track if the jth feature is used in the tth split:

$$-s_{jt} \leq a_{jt} \leq s_{jt}, \quad j = 1, \ldots, p, \ \forall t \in \mathcal{T}_B$$

We must also make sure that the values of s_{jt} and d_t are compatible. The following constraints ensure that $d_t = 1$ if and only if any variable is used in the split:

$$\begin{aligned} &s_{jt} \leq d_t, && j = 1, \ldots, p, \ \forall t \in \mathcal{T}_B, \\ &\sum_{j=1}^{p} s_{jt} \geq d_t, && \forall t \in \mathcal{T}_B. \end{aligned}$$

Finally, we modify the objective function to penalize the number of variables used across the splits in the tree:

$$\min \ \frac{1}{\hat{L}} \sum_{t \in \mathcal{T}_L} L_t + \ \alpha \cdot \sum_{t \in \mathcal{T}_B} \sum_{j=1}^{p} s_{jt}$$

Combining all of these changes yields the complete *Optimal Classification Trees with Hyperplanes* (OCT-H) model:

$$
\begin{aligned}
\min \quad & \frac{1}{\hat{L}} \sum_{t \in \mathcal{T}_L} L_t + \ \alpha \cdot \sum_{t \in \mathcal{T}_B} \sum_{j=1}^{p} s_{jt} && && (28)\\
\text{s.t.} \quad & L_t \geq N_t - N_{kt} - n(1 - c_{kt}), && k = 1, \ldots, K, \ \forall t \in \mathcal{T}_L,\\
& L_t \leq N_t - N_{kt} + n c_{kt}, && k = 1, \ldots, K, \ \forall t \in \mathcal{T}_L,\\
& L_t \geq 0, && \forall t \in \mathcal{T}_L,\\
& N_{kt} = \frac{1}{2} \sum_{i=1}^{n} (1 + Y_{ik}) z_{it}, && k = 1, \ldots, K, \ \forall t \in \mathcal{T}_L,\\
& N_t = \sum_{i=1}^{n} z_{it}, && \forall t \in \mathcal{T}_L,\\
& \sum_{k=1}^{K} c_{kt} = l_t, && \forall t \in \mathcal{T}_L,\\
& \mathbf{a}_m^T \mathbf{x}_i + \mu \leq b_m + (2 + \mu)(1 - z_{it}), && i = 1, \ldots, n, \ \forall t \in \mathcal{T}_B, \ \forall m \in A_L(t),\\
& \mathbf{a}_m^T \mathbf{x}_i \geq b_m - 2(1 - z_{it}), && i = 1, \ldots, n, \ \forall t \in \mathcal{T}_B, \ \forall m \in A_R(t),\\
& \sum_{t \in \mathcal{T}_L} z_{it} = 1, && i = 1, \ldots, n,\\
& z_{it} \leq l_t, && \forall t \in \mathcal{T}_L,\\
& \sum_{i=1}^{n} z_{it} \geq N_{\min} l_t, && \forall t \in \mathcal{T}_L,\\
& \sum_{j=1}^{p} \hat{a}_{jt} \leq d_t, && \forall t \in \mathcal{T}_B,\\
& \hat{a}_{jt} \geq a_{jt}, && j = 1, \ldots, p, \ \forall t \in \mathcal{T}_B,\\
& \hat{a}_{jt} \geq -a_{jt}, && j = 1, \ldots, p, \ \forall t \in \mathcal{T}_B,\\
& -s_{jt} \leq a_{jt} \leq s_{jt}, && j = 1, \ldots, p, \ \forall t \in \mathcal{T}_B\\
& s_{jt} \leq d_t, && j = 1, \ldots, p, \ \forall t \in \mathcal{T}_B,\\
& \sum_{j=1}^{p} s_{jt} \geq d_t, && \forall t \in \mathcal{T}_B,\\
& -d_t \leq b_t \leq d_t, && \forall t \in \mathcal{T}_B,\\
& d_t \leq d_{p(t)}, && \forall t \in \mathcal{T}_B \setminus \{1\},\\
& z_{it}, l_t \in \{0, 1\}, && i = 1, \ldots, n, \ \forall t \in \mathcal{T}_L,\\
& d_t \in \{0, 1\}, && j = 1, \ldots, p, \ \forall t \in \mathcal{T}_B.
\end{aligned}
$$

Note that the OCT-H formulation actually contains fewer binary variables than the OCT problem, as the integrality requirements on $\mathbf{a}_t$ have been relaxed. This means the OCT-H problem tends to be easier to solve than the OCT problem, which is not usually the case for hyperplane tree methods.

Additionally, from this formulation we can easily obtain the OCT problem as a special case by restoring the integrality constraints on $\mathbf{a}_t$. The close relationship between the univariate and multivariate problems reinforces the notion that the MIO formulation is the natural way to view the decision tree problem.

4.2 Warm Starts for Optimal Classification Trees with Hyperplanes

As a result of relaxing the constraints that required the splits in the tree be univariate, the warm start trees from before are no longer ideal for warm starting the solving process. Because we have only relaxed constraints in the model, these previous warm starts are still feasible solutions, but it is possible that we can inexpensively find better warm starts that exploit the increased solution space in the OCT-H problem. Recall that the quality of the warm start solution has a large effect on the speed of the solving process, so it is generally a good idea for us to generate higher quality warm starts where possible.

The approach we employ to generate a high-quality warm start to the OCT-H problem is to use greedy top-down induction with multivariate splits. This is a natural extension of the univariate heuristics to the multivariate case. To determine the multivariate split at each node, we simply solve a reduced version of the OCT-H problem for the points in the current node by restricting the solution to a single split. This is a much smaller problem than the full OCT-H problem, and so it solves much faster and without the need for a warm start, although it can also use the best univariate split as a good warm start.

Note that it would be possible to determine the split at each node using any other multivariate classification technique, such as logistic regression or support vector machines. This may increase the speed at which the multivariate warm starts are generated, since highly optimized versions of these algorithms exist for most programming languages. It may also lead to increased quality of the warm start solution, although this has not been tested.

4.3 Training Procedure for Optimal Classification Trees with Hyperplanes

The OCT-H problem (28) has exactly the same parameters as the OCT problem, and thus the procedure in Section 3 can be used for tuning these parameters. The only difference is that each split in the tree can now use up to p variables, and so the maximum tree complexity, $C_{\max}$, becomes $p(2^D - 1)$. As before, we can search over all values of $C = 1, \ldots, C_{\max}$, solving the complexity-constrained problem for each and then post-processing to obtain the corresponding tuned value for α.

It can happen that p is too large to make this computationally feasible, and it may be more efficient to simply search over a discretized range of possible α values as is typically done during validation. In this case, the number of problems to solve is $D \cdot |\mathcal{A}|$, where $\mathcal{A}$ is the discretized set of values to search. The same approach of reusing the solutions from lower depths as warm starts can still be applied to speed up the solutions. Whether this approach is better than the constraint-based approach depends on the size of $\mathcal{A}$ that is chosen, and the number of problems that must be solved by each approach, which is $D \cdot |\mathcal{A}|$ compared to $p(2^D - 1)$.

5 Computational Experiments with Synthetic Datasets

In this section, we examine the performance of Optimal Classification Trees on a variety of synthetically-generated datasets in order to understand how effective the method is at discovering the underlying ground truth in a dataset whose structure is in fact described by a decision tree.

The experiments we conduct are adaptations of the experiments by Murthy and Salzberg (1995b). These experiments use a single decision tree as the ground truth to generate datasets, and then different methods are used to induce decision trees on these datasets and are then compared to the ground truth tree to evaluate performance. To construct the ground truth, a decision tree of a specified depth is created by choosing splits at random, and the leaves of the resulting tree are labeled such that no two adjacent leaves have the same label, ensuring that this tree is the smallest representation of the ground truth. The training and test datasets are created by generating each $\mathbf{x}_i$ as a uniform random vector, and then using the ground truth tree to assign the corresponding label to the point.

Following Murthy and Salzberg (1995b), we report six measures of tree quality in all experiments:

- In-sample accuracy: accuracy on the training set.
- Out-of-sample accuracy: accuracy on the test set.
- Tree size: number of leaf nodes.
- Maximum depth: total depth of the tree.
- Average depth: average depth of leaf nodes in the tree.
- Expected depth: average depth of leaf nodes in the tree, weighted by the proportion of the test set that belongs to each leaf node (this is also sometimes referred to as *dynamic complexity*).

In each experiment, 10 random trees are generated as different ground truths, and 10 training and test set pairs are created using each tree. The size of the training set is specified by the experiment, and the size of the test set is always $2^D \cdot (p-1) \cdot 500$, where D is the depth of the ground truth tree and p is the number of features in the data, both specified by the particular experiment. The quality measures are calculated over all 100 instances of the problem, and we present the means and standard errors of these measures.

The methods we compare are our OCT method, the standard CART heuristic, and a modified CART heuristic that imposes a depth limit equal to the depth of the ground truth. The purpose of this third method is to examine the impact of trimming the CART solution as a method of obtaining a simpler and more interpretable tree. For experiments with noise in the data, 10% of the training set is reserved for validation purposes in order to tune the complexity parameter α. Note that Murthy and Salzberg (1995b) found that C4.5 and CART gave near-identical results for all experiments they conducted and reported only the C4.5 results, so our results for CART should also be similar to those that would be obtained using C4.5.

The first experiment evaluates the effectiveness of each method as the problem complexity increases by increasing the depth of the ground truth while holding the training set size and dimension fixed at $n = 100$ and $p = 2$, respectively. There is no noise present in the training data. Table 1 shows the results of this experiment. We see that depth-constrained CART is significantly worse than the

True Depth	Method	In-sample Accuracy	Out-of-sample Accuracy	Tree Size	Depth		
					Maximum	Average	Expected
2	CART-depth	86.60 ± 2.01	92.03 ± 1.01	3.3 ± 0.1	1.8 ± 0.0	1.7 ± 0.0	1.6 ± 0.0
	CART	100.00 ± 0.00	97.79 ± 0.18	4.6 ± 0.2	2.8 ± 0.1	2.2 ± 0.1	2.0 ± 0.1
	OCT	100.00 ± 0.00	98.11 ± 0.14	3.9 ± 0.0	2.0 ± 0.0	2.0 ± 0.0	1.9 ± 0.0
	Ground truth	100.00 ± 0.00	100.00 ± 0.00	3.9 ± 0.0	2.0 ± 0.0	2.0 ± 0.0	1.9 ± 0.0
3	CART-depth	88.21 ± 1.32	93.43 ± 0.42	5.2 ± 0.1	3.0 ± 0.0	2.5 ± 0.0	2.3 ± 0.0
	CART	99.96 ± 0.04	95.43 ± 0.24	7.6 ± 0.3	4.5 ± 0.1	3.3 ± 0.1	2.7 ± 0.1
	OCT	100.00 ± 0.00	96.01 ± 0.20	7.4 ± 0.1	3.0 ± 0.0	2.9 ± 0.0	2.9 ± 0.0
	Ground truth	100.00 ± 0.00	100.00 ± 0.00	7.8 ± 0.0	3.0 ± 0.0	3.0 ± 0.0	2.9 ± 0.0
4	CART-depth	90.49 ± 0.86	87.36 ± 0.57	9.5 ± 0.2	4.0 ± 0.0	3.4 ± 0.0	3.1 ± 0.0
	CART	100.00 ± 0.00	89.82 ± 0.39	12.9 ± 0.3	5.5 ± 0.1	4.1 ± 0.1	3.5 ± 0.1
	OCT	100.00 ± 0.00	89.91 ± 0.33	14.7 ± 0.1	4.0 ± 0.0	3.9 ± 0.0	3.9 ± 0.0
	Ground truth	100.00 ± 0.00	100.00 ± 0.00	15.1 ± 0.1	4.0 ± 0.0	3.9 ± 0.0	3.9 ± 0.0

Table 1 Effects of noise in ground truth depth. No noise in data. Training size = 100.

Training set size	Method	In-sample Accuracy	Out-of-sample Accuracy	Tree Size	Depth		
					Maximum	Average	Expected
100	CART-depth	86.60 ± 2.01	92.03 ± 1.01	3.3 ± 0.1	1.8 ± 0.0	1.7 ± 0.0	1.6 ± 0.0
	CART	100.00 ± 0.00	97.79 ± 0.18	4.6 ± 0.2	2.8 ± 0.1	2.2 ± 0.1	2.0 ± 0.1
	OCT	100.00 ± 0.00	98.16 ± 0.13	3.9 ± 0.0	2.0 ± 0.0	1.9 ± 0.0	1.9 ± 0.0
	Ground truth	100.00 ± 0.00	100.00 ± 0.00	3.9 ± 0.0	2.0 ± 0.0	2.0 ± 0.0	1.9 ± 0.0
200	CART-depth	93.48 ± 1.03	97.18 ± 0.47	3.1 ± 0.1	1.9 ± 0.0	1.7 ± 0.0	1.5 ± 0.0
	CART	100.00 ± 0.00	99.24 ± 0.06	3.7 ± 0.1	2.5 ± 0.1	2.0 ± 0.0	1.7 ± 0.0
	OCT	100.00 ± 0.00	99.21 ± 0.07	3.8 ± 0.0	2.0 ± 0.0	1.9 ± 0.0	1.9 ± 0.0
	Ground truth	100.00 ± 0.00	100.00 ± 0.00	3.7 ± 0.0	2.0 ± 0.0	1.9 ± 0.0	1.8 ± 0.0
400	CART-depth	89.62 ± 1.49	97.58 ± 0.25	3.3 ± 0.0	2.0 ± 0.0	1.8 ± 0.0	1.6 ± 0.0
	CART	100.00 ± 0.00	99.47 ± 0.04	4.7 ± 0.1	2.7 ± 0.0	2.3 ± 0.0	1.9 ± 0.0
	OCT	100.00 ± 0.00	99.64 ± 0.03	3.9 ± 0.0	2.0 ± 0.0	2.0 ± 0.0	2.0 ± 0.0
	Ground truth	100.00 ± 0.00	100.00 ± 0.00	3.8 ± 0.0	2.0 ± 0.0	1.9 ± 0.0	1.9 ± 0.0
800	CART-depth	89.75 ± 1.51	96.59 ± 0.52	3.2 ± 0.0	2.0 ± 0.0	1.7 ± 0.0	1.6 ± 0.0
	CART	100.00 ± 0.00	99.80 ± 0.02	4.0 ± 0.1	2.4 ± 0.1	2.1 ± 0.0	1.8 ± 0.0
	OCT	100.00 ± 0.00	99.82 ± 0.02	3.8 ± 0.0	2.0 ± 0.0	1.9 ± 0.0	1.9 ± 0.0
	Ground truth	100.00 ± 0.00	100.00 ± 0.00	3.6 ± 0.0	2.0 ± 0.0	1.9 ± 0.0	1.9 ± 0.0
1600	CART-depth	97.38 ± 0.57	99.65 ± 0.05	3.4 ± 0.0	2.0 ± 0.0	1.8 ± 0.0	1.6 ± 0.0
	CART	100.00 ± 0.00	99.89 ± 0.01	3.9 ± 0.1	2.3 ± 0.1	2.0 ± 0.0	1.7 ± 0.0
	OCT	100.00 ± 0.00	99.89 ± 0.01	3.8 ± 0.0	2.0 ± 0.0	1.9 ± 0.0	1.9 ± 0.0
	Ground truth	100.00 ± 0.00	100.00 ± 0.00	3.5 ± 0.1	2.0 ± 0.0	1.8 ± 0.0	1.8 ± 0.0

Table 2 Effects of training set size. No noise in data. Ground truth trees are depth 2.

other methods when compared to the ground truth, a result that is common to all of these experiments. In terms of out-of-sample accuracy, both CART and OCT are about the same in all tests, with OCT having a very slight advantage at all depths that is largest at depth 3. We see that the out-of-sample accuracies decrease significantly as the depth increases, which is to be expected because the training problem is becoming more complex with the same amount of training data available. In terms of the other quality measures, OCT matches the ground truth nearly identically in all cases across the measures, whereas CART is significantly different and seems to be learning trees that are structurally different to the ground truth. This experiment demonstrates that although different methods can have the same out-of-sample accuracy, the underlying structure learned by each method can still be significantly different from one another, and in this case OCT is a much closer match for the ground truth.

Class Noise	Method	In-sample Accuracy	Out-of-sample Accuracy	Tree Size	Depth		
					Maximum	Average	Expected
0%	CART-depth	98.77 ± 0.21	98.27 ± 0.19	3.4 ± 0.0	2.0 ± 0.0	1.8 ± 0.0	1.7 ± 0.0
	CART	100.00 ± 0.00	99.03 ± 0.08	4.0 ± 0.1	2.4 ± 0.1	2.0 ± 0.0	1.8 ± 0.0
	OCT	100.00 ± 0.00	99.16 ± 0.07	3.9 ± 0.0	2.0 ± 0.0	2.0 ± 0.0	2.0 ± 0.0
	Ground truth	100.00 ± 0.00	100.00 ± 0.00	3.6 ± 0.0	2.0 ± 0.0	1.9 ± 0.0	1.8 ± 0.0
5%	CART-depth	88.02 ± 0.75	95.96 ± 0.67	3.5 ± 0.1	1.9 ± 0.0	1.8 ± 0.0	1.8 ± 0.0
	CART	91.20 ± 0.20	98.62 ± 0.14	3.8 ± 0.1	2.2 ± 0.1	1.9 ± 0.0	1.9 ± 0.0
	OCT	91.46 ± 0.16	98.84 ± 0.11	3.7 ± 0.1	1.9 ± 0.0	1.9 ± 0.0	1.9 ± 0.0
	Ground truth	95.16 ± 0.03	100.00 ± 0.00	3.8 ± 0.0	2.0 ± 0.0	1.9 ± 0.0	2.0 ± 0.0
10%	CART-depth	79.77 ± 0.60	95.39 ± 0.49	3.5 ± 0.1	2.0 ± 0.0	1.8 ± 0.0	1.9 ± 0.0
	CART	83.71 ± 0.45	98.33 ± 0.23	4.3 ± 0.2	2.6 ± 0.1	2.2 ± 0.1	2.0 ± 0.0
	OCT	83.97 ± 0.26	98.85 ± 0.10	3.9 ± 0.0	2.0 ± 0.0	1.9 ± 0.0	2.0 ± 0.0
	Ground truth	90.66 ± 0.06	100.00 ± 0.00	3.9 ± 0.0	2.0 ± 0.0	2.0 ± 0.0	2.0 ± 0.0
15%	CART-depth	74.91 ± 0.51	97.90 ± 0.26	3.4 ± 0.1	1.9 ± 0.0	1.8 ± 0.0	1.7 ± 0.0
	CART	76.03 ± 0.52	97.70 ± 0.33	4.0 ± 0.2	2.3 ± 0.1	1.9 ± 0.1	1.8 ± 0.1
	OCT	76.18 ± 0.33	98.53 ± 0.21	3.5 ± 0.1	1.9 ± 0.0	1.8 ± 0.0	1.8 ± 0.0
	Ground truth	86.50 ± 0.10	100.00 ± 0.00	3.6 ± 0.0	2.0 ± 0.0	1.9 ± 0.0	1.8 ± 0.0
20%	CART-depth	66.17 ± 0.66	95.01 ± 0.44	3.5 ± 0.1	1.9 ± 0.0	1.8 ± 0.0	1.8 ± 0.0
	CART	68.39 ± 0.70	95.43 ± 0.46	4.4 ± 0.3	2.6 ± 0.2	2.1 ± 0.1	1.9 ± 0.1
	OCT	68.53 ± 0.43	97.34 ± 0.28	3.5 ± 0.1	1.9 ± 0.0	1.8 ± 0.0	1.8 ± 0.0
	Ground truth	82.93 ± 0.15	100.00 ± 0.00	3.7 ± 0.0	2.0 ± 0.0	1.9 ± 0.0	1.9 ± 0.0
25%	CART-depth	62.02 ± 0.56	95.95 ± 0.49	3.5 ± 0.1	1.9 ± 0.0	1.8 ± 0.0	1.9 ± 0.0
	CART	63.34 ± 0.60	96.20 ± 0.38	4.2 ± 0.4	2.4 ± 0.2	2.0 ± 0.1	2.0 ± 0.1
	OCT	62.69 ± 0.57	96.86 ± 0.38	3.6 ± 0.1	1.9 ± 0.0	1.8 ± 0.0	1.9 ± 0.0
	Ground truth	79.10 ± 0.18	100.00 ± 0.00	3.9 ± 0.0	2.0 ± 0.0	2.0 ± 0.0	2.0 ± 0.0

Table 3 Effects of noise in class labels. Ground truth trees are depth 2.

The second experiment investigates the effect of the amount of available training data relative to the complexity of the problem. We increase the size of the training set while holding the depth of the ground truth fixed at $D = 2$ and the dimension fixed at $p = 2$. There is no noise in the training data. Table 2 shows a summary of the results. We see that all methods increase in out-of-sample accuracy as the training size increases, approaching 100% accuracy at the higher sizes. We see that initially depth-constrained CART is a poor approximation to the ground truth, but it becomes as accurate out-of-sample as the other methods if sufficient training data is available. In a data-poor environment, depth-constrained CART performs significantly worse, and OCT offers a slight improvement over CART in terms of out-of-sample accuracy. This offers clear evidence against the notion that optimal methods tend to overfit the training data in data-poor scenarios; in fact the optimal method performs stronger. In terms of the other metrics, OCT most closely matches the attributes of the true trees in nearly all cases, indicating it is actually learning the truth in the data.

The third experiment examines the impact of adding noise to the labels of the training data. In this experiment, 100 training points were generated per distinct label of the ground truth tree ($D = 2$, $p = 2$). Noise was then added to the training set by increasing by 1 the label of a random k% of the points, where k is a parameter of the experiment. Table 3 shows the impact of increasing levels of label perturbations in the training data. As the level of noise increases, the accuracies of all methods tend to decrease, as would be expected. The difference between the out-of-sample accuracies of CART and OCT increases with the level of noise up to 20%, indicating that OCT is able to better identify the true tree than CART when exposed to such noise, and this difference quickly becomes significant

Attribute Noise	Method	In-sample Accuracy	Out-of-sample Accuracy	Tree Size	Depth		
					Maximum	Average	Expected
0%	CART-depth	98.77 ± 0.21	98.27 ± 0.19	3.4 ± 0.0	2.0 ± 0.0	1.8 ± 0.0	1.7 ± 0.0
	CART	100.00 ± 0.00	99.03 ± 0.08	4.0 ± 0.1	2.4 ± 0.1	2.0 ± 0.0	1.8 ± 0.0
	OCT	100.00 ± 0.00	99.16 ± 0.07	3.9 ± 0.0	2.0 ± 0.0	2.0 ± 0.0	2.0 ± 0.0
	Ground truth	100.00 ± 0.00	100.00 ± 0.00	3.6 ± 0.0	2.0 ± 0.0	1.9 ± 0.0	1.8 ± 0.0
5%	CART-depth	95.13 ± 0.67	96.46 ± 0.54	3.4 ± 0.1	1.9 ± 0.0	1.8 ± 0.0	1.7 ± 0.0
	CART	98.77 ± 0.10	98.76 ± 0.13	3.9 ± 0.1	2.3 ± 0.1	2.0 ± 0.0	2.0 ± 0.0
	OCT	98.83 ± 0.10	99.10 ± 0.07	3.6 ± 0.1	2.0 ± 0.0	1.9 ± 0.0	1.9 ± 0.0
	Ground truth	99.28 ± 0.06	100.00 ± 0.00	3.8 ± 0.0	2.0 ± 0.0	1.9 ± 0.0	1.9 ± 0.0
10%	CART-depth	93.77 ± 0.82	95.19 ± 0.69	3.1 ± 0.1	1.9 ± 0.0	1.7 ± 0.0	1.5 ± 0.0
	CART	98.02 ± 0.17	98.78 ± 0.12	3.6 ± 0.1	2.4 ± 0.0	1.9 ± 0.0	1.6 ± 0.0
	OCT	98.03 ± 0.18	98.80 ± 0.14	3.4 ± 0.0	2.0 ± 0.0	1.8 ± 0.0	1.7 ± 0.0
	Ground truth	98.73 ± 0.10	100.00 ± 0.00	3.4 ± 0.0	2.0 ± 0.0	1.8 ± 0.0	1.7 ± 0.0
15%	CART-depth	93.87 ± 0.76	97.08 ± 0.36	3.0 ± 0.0	1.9 ± 0.0	1.6 ± 0.0	1.5 ± 0.0
	CART	96.49 ± 0.26	97.78 ± 0.25	3.3 ± 0.1	2.2 ± 0.1	1.8 ± 0.0	1.6 ± 0.0
	OCT	96.48 ± 0.26	98.05 ± 0.22	3.2 ± 0.1	1.9 ± 0.0	1.7 ± 0.0	1.6 ± 0.0
	Ground truth	97.37 ± 0.20	100.00 ± 0.00	3.3 ± 0.0	2.0 ± 0.0	1.8 ± 0.0	1.6 ± 0.0
20%	CART-depth	92.07 ± 0.84	96.92 ± 0.43	2.9 ± 0.1	1.7 ± 0.0	1.5 ± 0.0	1.5 ± 0.0
	CART	95.56 ± 0.29	97.82 ± 0.22	3.3 ± 0.1	2.1 ± 0.1	1.7 ± 0.1	1.6 ± 0.0
	OCT	95.73 ± 0.27	97.81 ± 0.22	3.2 ± 0.1	1.8 ± 0.0	1.7 ± 0.0	1.7 ± 0.0
	Ground truth	97.16 ± 0.19	100.00 ± 0.00	3.7 ± 0.0	2.0 ± 0.0	1.9 ± 0.0	1.9 ± 0.0
25%	CART-depth	91.47 ± 0.56	95.09 ± 0.54	3.2 ± 0.0	2.0 ± 0.0	1.7 ± 0.0	1.7 ± 0.0
	CART	94.28 ± 0.29	97.14 ± 0.27	3.6 ± 0.1	2.4 ± 0.1	1.9 ± 0.0	1.8 ± 0.0
	OCT	94.51 ± 0.28	97.29 ± 0.25	3.6 ± 0.1	2.0 ± 0.0	1.9 ± 0.0	1.9 ± 0.0
	Ground truth	96.01 ± 0.23	100.00 ± 0.00	3.7 ± 0.0	2.0 ± 0.0	1.9 ± 0.0	1.8 ± 0.0

Table 4 Effects of noise in attribute values. Ground truth trees are depth 2.

(1–2% for noise levels of 15–20%). At the highest level of noise, the difference between these methods is diminished, reflecting the difficulty of learning the truth in such a noisy environment. The other metrics also indicate that OCT again most closely matches the ground truth as the noise levels are increased. We remark that these results counter the idea that optimal methods are less robust to noise in the data than current heuristic approaches due to problems of overfitting, as we clearly see that the OCT trees are significantly stronger out-of-sample than the heuristic counterparts for high levels of label noise.

The fourth experiment is similar to the previous one, except it considers noise in the features of the training data instead of the labels. Again, 100 training points are generated for each distinct label in the ground truth tree ($D = 2$, $p = 2$). We introduce noise by selecting a random $k\%$ of the training points and to each of these adding random noise $\epsilon \sim U(-0.1, 0.1)$ to every feature. Table 4 shows the impact of increasing levels of this feature noise in the training data. At all levels of noise, the out-of-sample accuracies of CART and OCT are comparable, as are the other tree metrics. Even at the highest level of noise, where 25% of the training data are perturbed in all features by up to 10%, both methods have an out-of-sample accuracy above 97%. We conclude that in the presence of significant feature noise, neither method outperforms the other, and the optimal method is clearly not overfitting the training data.

The final experiment considers the effect of increasing the dimensionality of the problem while holding the training size and ground truth depth fixed at $n = 100$ and $D = 2$, respectively. Table 5 shows the effect of increasing the number of features for fixed training size and tree depth. At the lower dimensions, there is no significant difference between CART and OCT, but at higher dimensions the

Number of features	Method	In-sample Accuracy	Out-of-sample Accuracy	Tree Size	Depth		
					Maximum	Average	Expected
2	CART-depth	86.60 ± 2.01	92.03 ± 1.01	3.3 ± 0.1	1.8 ± 0.0	1.7 ± 0.0	1.6 ± 0.0
	CART	100.00 ± 0.00	97.79 ± 0.18	4.6 ± 0.2	2.8 ± 0.1	2.2 ± 0.1	2.0 ± 0.1
	OCT	100.00 ± 0.00	98.16 ± 0.13	3.9 ± 0.0	2.0 ± 0.0	1.9 ± 0.0	1.9 ± 0.0
	Ground truth	100.00 ± 0.00	100.00 ± 0.00	3.9 ± 0.0	2.0 ± 0.0	2.0 ± 0.0	1.9 ± 0.0
4	CART-depth	98.31 ± 0.37	98.25 ± 0.13	3.3 ± 0.0	2.0 ± 0.0	1.8 ± 0.0	1.6 ± 0.0
	CART	100.00 ± 0.00	98.55 ± 0.11	3.6 ± 0.1	2.3 ± 0.0	1.9 ± 0.0	1.6 ± 0.0
	OCT	100.00 ± 0.00	98.48 ± 0.10	3.8 ± 0.0	2.0 ± 0.0	1.9 ± 0.0	1.9 ± 0.0
	Ground truth	100.00 ± 0.00	100.00 ± 0.00	3.6 ± 0.0	2.0 ± 0.0	1.9 ± 0.0	1.8 ± 0.0
8	CART-depth	95.99 ± 0.70	96.37 ± 0.42	3.8 ± 0.0	2.0 ± 0.0	1.9 ± 0.0	1.9 ± 0.0
	CART	100.00 ± 0.00	97.64 ± 0.18	4.5 ± 0.1	2.5 ± 0.1	2.2 ± 0.0	2.0 ± 0.0
	OCT	100.00 ± 0.00	98.25 ± 0.12	4.0 ± 0.0	2.0 ± 0.0	2.0 ± 0.0	2.0 ± 0.0
	Ground truth	100.00 ± 0.00	100.00 ± 0.00	3.9 ± 0.0	2.0 ± 0.0	2.0 ± 0.0	2.0 ± 0.0
16	CART-depth	96.33 ± 0.59	97.28 ± 0.20	3.9 ± 0.0	2.0 ± 0.0	2.0 ± 0.0	2.0 ± 0.0
	CART	100.00 ± 0.00	97.42 ± 0.17	4.4 ± 0.1	2.4 ± 0.0	2.2 ± 0.0	2.0 ± 0.0
	OCT	100.00 ± 0.00	98.06 ± 0.16	4.0 ± 0.0	2.0 ± 0.0	2.0 ± 0.0	2.0 ± 0.0
	Ground truth	100.00 ± 0.00	100.00 ± 0.00	4.0 ± 0.0	2.0 ± 0.0	2.0 ± 0.0	2.0 ± 0.0

Table 5 Effects of dimensionality. Training set size = 100. No noise. Ground truth trees are depth 2.

difference is about 0.6% and is significant. The other metrics show that again the OCT trees are the best match for the ground truth. These results echo those from the second test and show that the optimal method performs stronger relative to CART in data-poor environments where relatively little training data is available compared to the problem complexity.

We conclude by summarizing the collective findings of these tests with synthetic data. In nearly all cases, the trees generated by the OCT method most closely matched the quality metrics of the ground truth trees. In particular, the OCT trees in most cases had out-of-sample accuracies that were significantly higher than those of CART, even in the presence of significant levels of noise in the training data. This provides strong evidence against the widely-held belief that optimal methods are more inclined to overfit the training set at the expense of out-of-sample accuracy. A secondary conclusion is that the depth-constrained CART method is a poor way of constructing a decision tree of a given depth. Creating complete trees and then trimming these to shorter depths yields trees that are significantly worse in nearly all cases than those created by OCT, which creates the tree with the depth limit in mind.

6 Computational Results with Real-world Datasets

In this section, we report the performance of Optimal Classification Trees with and without hyperplanes on several publicly available datasets widely-used within the statistics and machine learning communities, and compare these to the prominent decision tree method, CART.

In order to provide a comprehensive benchmark of performance for optimal trees, we used a collection of 53 datasets obtained from the UCI Machine Learning Repository (Lichman 2013). These datasets are used regularly for reporting and comparing the performance of different classification methods. The datasets we use have sizes up to thousands of points, which we demonstrate can be handled by our methods. Datasets with tens of thousands of points were tested, but the

time required for our methods to generate high-quality solutions was prohibitive. For these experiments, we seek to demonstrate the improvement delivered by our methods on these common datasets of manageable size.

Each dataset was split into three parts: the training set (50%), the validation set (25%), and the testing set (25%). The training and validation sets were used to tune the value of the hyperparameter α. We then used this α to train a tree on the combined training and validation sets, which we then evaluate against the testing set to determine the out-of-sample accuracy.

To reduce the amount of computation required across all the datasets, we elected not to tune the parameter $N_{\min}$. Instead, we set this to 5% of the total number of points in the dataset so that this value would work across all datasets.

We trained Optimal Trees of depths 1–4 on all datasets, and compared them to solutions from CART trimmed to the same depth. These CART solutions were obtained by fixing $N_{\min}$ to the same value as for Optimal Trees, and tuning the hyperparameter α in the usual way for CART. The resulting tree was then trimmed to the specified depth. This lets us evaluate the performance of all methods in the same depth-constrained scenario, which is often important for applications that require interpretability of the solutions. In such cases, we seek solutions with very few splits that best capture the structure in the data. We collected data for CART without the depth constraints to examine whether shallower Optimal Trees can achieve the same accuracy as deep CART trees.

In order to minimize the effect of the particular splitting of the data into training, validation and testing sets, the entire process was conducted five times for each dataset, with a different splitting each time. The final out-of-sample accuracies were then obtained by averaging the results across these five runs.

For testing the performance of CART, we used the RPART package (Therneau et al 2015) in the R programming language (R Core Team 2015). The hyperparameters $N_{\min}$ and α correspond to the parameters `minbucket` and `cp`. As discussed earlier, we post-processed the trees to satisfy any depth constraints.

The Optimal Classification Tree problems (OCT and OCT-H) were implemented in the JULIA programming language, which is a rapidly maturing language designed for high-performance scientific computing (Bezanson et al 2014). We formulated our MIO models using the JuMP package, a state-of-the-art library for algebraic modeling and mathematical optimization (Lubin and Dunning 2015), and solved them using the GUROBI 6.5 solver, one of the fastest available commercial MIO solvers (Gurobi Optimization Inc. 2015b). Again, we tuned the value of α through the validation process described in Section 3.

All problems were solved in parallel using the Amazon Web Services (AWS) Elastic Compute Cloud (EC2). We used the compute-optimized `c4.xlarge` instances, which have have 4 virtual CPU cores (2.9 GHz) and 7.5 GB RAM.

A time limit was applied to each Optimal Tree problem. On most datasets for OCT this was 30 minutes, but for some of the larger datasets we increased the time limit up to 2 hours to ensure that the solver could make progress. The OCT-H problem is easier to solve and so only required time limits of 5–15 minutes. When generating heuristic warm starts for the multivariate problem by recursively solving the OCT-H problem with a single split, we used a time limit of 60 seconds. We found this was more than enough to find a good greedy solution.

As discussed in Section 2, the application of a time limit when solving an MIO problem when using strong warm starts and heuristics does not usually affect the

Max. depth	CART	OCT	OCT-H (top-down)	OCT-H
1	71.0%	71.3%	74.5%	76.1%
2	76.2%	78.0%	78.7%	81.0%
3	78.8%	79.5%	79.9%	82.3%
4	79.8%	80.4%	80.6%	82.9%

Table 6 Mean out-of-sample accuracies across all datasets for each method, according to the maximum tree depth.

quality of the solution if the time limit is sufficiently large. As we saw in Figure 3, the solution that is eventually shown to be optimal is typically found very close to the beginning of the solve, especially when warm starts are used. Moreover, the majority of the time is spent in proving that this solution is in fact optimal, which is not required for our purposes. If we set the time limit intelligently, we can avoid the lengthy process of proving optimality while still generating high quality solutions. However, this does mean that most of our results for OCT and OCT-H do not carry explicit certificates of optimality.

6.1 Overall Comparison of All Methods

We first provide a summary of all methods for comparison purposes. Table 6 shows the average out-of-sample accuracy across all datasets for each method and tree depth. For reference we also include the results for a top-down induction version of OCT-H, where the tree is decided greedily, one split at a time, providing a reference point for the performance of top-down multivariate methods. We see that CART performs the weakest, and is outperformed at all depths by OCT by about 1–2%, demonstrating the advantage that can be realized by using optimal trees. OCT-H is significantly more powerful than all other methods; in particular, OCT-H performs significantly stronger than the top-down OCT-H heuristic, over 2% better on average, indicating that multivariate trees also benefit significantly from taking a global perspective when choosing the splits.

When we run CART without constraining the depth of the trees, the maximum tree depth required is 10 and the resulting average accuracy across all datasets is 80.7% . We note this is very close to the accuracy of 80.4% of OCT at depth 4. It is quite possible that with a longer time limit, OCT at depth 4 would be able to achieve accuracies similar to those of CART at depth 10. This would be very significant for applications that require interpretation of the decision trees, since shallow trees are much more interpretable. We also note that even the depth 2 accuracy of OCT-H at 81.0% is greater than this depth 10 CART accuracy, indicating that very shallow multivariate decision trees can outperform the deepest univariate trees.

6.2 Optimal Classification Trees vs. CART

We now present an in-depth direct comparison of OCT and CART. Both methods seek to solve the same decision tree problem, so this comparison aims to show the

Dataset				Mean out-of-sample accuracy		Mean improvement
Name	n	p	K	CART	OCT	
acute-inflammations-1	120	6	2	88.7	**100.0**	$+11.33 \pm 1.70$
acute-inflammations-2	120	6	2	**100.0**	**100.0**	0.00 ± 0.00
balance-scale	625	4	3	64.5	**67.1**	$+2.68 \pm 1.30$
banknote-authentication	1372	4	2	89.0	**90.1**	$+1.05 \pm 0.77$
blood-transfusion	748	4	2	**75.5**	**75.5**	0.00 ± 0.00
breast-cancer-diagnostic	569	30	2	90.5	**91.9**	$+1.40 \pm 0.63$
breast-cancer-prognostic	194	32	2	**75.5**	75.1	-0.41 ± 0.41
breast-cancer	683	9	2	92.3	**94.5**	$+2.22 \pm 1.00$
car-evaluation	1728	15	4	**73.7**	**73.7**	0.00 ± 0.00
chess-king-rook-vs-king-pawn	3196	37	2	78.2	**86.7**	$+8.51 \pm 0.56$
climate-model-crashes	540	18	2	90.1	**90.5**	$+0.44 \pm 0.44$
congressional-voting-records	232	16	2	**98.6**	**98.6**	0.00 ± 0.00
connectionist-bench-sonar	208	60	2	70.4	**71.2**	$+0.77 \pm 0.77$
connectionist-bench	990	10	11	24.8	**29.0**	$+4.21 \pm 1.68$
contraceptive-method-choice	1473	11	3	46.8	**48.4**	$+1.57 \pm 1.57$
credit-approval	653	37	2	**87.7**	**87.7**	0.00 ± 0.00
cylinder-bands	277	484	2	**62.6**	**62.6**	0.00 ± 0.00
dermatology	358	34	6	65.4	**67.4**	$+2.02 \pm 2.02$
echocardiogram	61	6	2	74.7	**77.3**	$+2.67 \pm 1.63$
fertility	100	12	2	**88.0**	85.6	-2.40 ± 2.40
haberman-survival	306	3	2	**73.2**	**73.2**	0.00 ± 0.00
hayes-roth	132	4	3	**52.7**	45.5	-7.27 ± 3.66
heart-disease-cleveland	297	18	5	**54.1**	53.6	-0.53 ± 0.53
hepatitis	80	19	2	**83.0**	**83.0**	0.00 ± 0.00
image-segmentation	210	19	7	38.9	**53.6**	$+14.72 \pm 1.25$
indian-liver-patient	579	10	2	**71.7**	70.9	-0.83 ± 0.83
ionosphere	351	34	2	**87.8**	**87.8**	-0.00 ± 0.36
iris	150	4	3	**92.4**	**92.4**	0.00 ± 0.00
mammographic-mass	830	10	2	**81.2**	**81.2**	0.00 ± 0.00
monks-problems-1	124	11	2	57.4	**67.7**	$+10.32 \pm 6.24$
monks-problems-2	169	11	2	**60.9**	60.0	-0.93 ± 0.93
monks-problems-3	122	11	2	**94.2**	**94.2**	0.00 ± 0.00
optical-recognition	3823	64	10	**29.7**	29.4	-0.27 ± 0.17
ozone-level-detection-eight	1847	72	2	**93.1**	**93.1**	0.00 ± 0.00
ozone-level-detection-one	1848	72	2	**96.8**	**96.8**	0.00 ± 0.00
parkinsons	195	21	2	**84.1**	83.7	-0.41 ± 2.27
pima-indians-diabetes	768	8	2	71.9	**72.9**	$+1.04 \pm 1.04$
planning-relax	182	12	2	**71.1**	**71.1**	0.00 ± 0.00
qsar-biodegradation	1055	41	2	**76.4**	76.1	-0.30 ± 0.28
seeds	210	7	3	87.2	**88.7**	$+1.51 \pm 0.92$
seismic-bumps	2584	20	2	**93.3**	**93.3**	0.00 ± 0.00
soybean-small	47	37	4	72.7	**98.2**	$+25.45 \pm 1.82$
spambase	4601	57	2	84.2	**84.3**	$+0.03 \pm 0.03$
spect-heart	80	22	2	64.0	**65.0**	$+1.00 \pm 7.14$
spectf-heart	80	44	2	69.0	**72.0**	$+3.00 \pm 3.39$
statlog-project-german-credit	1000	48	2	70.1	**70.5**	$+0.40 \pm 0.40$
statlog-project-landsat-satellite	4435	36	6	**63.2**	**63.2**	0.00 ± 0.24
teaching-assistant-evaluation	151	52	3	38.9	**42.2**	$+3.24 \pm 2.76$
thoracic-surgery	470	24	2	**85.5**	84.6	-0.85 ± 0.85
thyroid-disease-ann-thyroid	3772	21	3	**95.6**	**95.6**	0.00 ± 0.00
thyroid-disease-new-thyroid	215	5	3	91.3	**92.8**	$+1.51 \pm 1.10$
tic-tac-toe-endgame	958	18	2	68.5	**69.6**	$+1.17 \pm 0.82$
wall-following-robot-2	5456	2	4	**94.0**	**94.0**	0.00 ± 0.00
wine	178	13	3	81.3	**91.6**	$+10.22 \pm 1.51$

Table 7 Full results for CART and OCT at depth 2.

Max. depth	CART wins	OCT wins	Ties	Accuracy improvement	p-value
1	4	19	30	0.36%	0.0141
2	9	25	19	1.86%	$\sim 10^{-6}$
3	14	21	18	0.73%	0.0065
4	13	21	19	0.62%	0.0279

Table 8 Comparison of CART and OCT across a range of depths, showing the number of datasets for which each method had the highest out-of-sample accuracy, and the mean improvement in out-of-sample accuracy when using OCT across all datasets along with the p-value indicating the statistical significance of this difference.

effectiveness of formulating the problem under an MIO framework and solving the exact same problem to optimality.

The entire set of mean out-of-sample accuracies on each dataset for both methods at depth 2 is provided in Table 7, which reports the size and dimension of the dataset, the number of class labels K, and the average out-of-sample accuracy of each method along with the mean accuracy improvement for OCT over CART and the corresponding standard error (results for depths 3 and 4 are provided in Appendix A).

In Table 8, we present the number of datasets for which each method performed strongest, broken down by the maximum depth of the trees. We see that at all depths, OCT is stronger on roughly twice as many datasets as CART. This table also shows the mean difference between the out-of-sample accuracies of each method across all datasets at each depth, along with the p-value indicating the statistical significance of this difference. We see that the OCT trees have a higher accuracy of around 0.5–2% at all depths, and that this difference is statistically significant for all depths. The largest difference is at depth 2, and at depths 3 and 4 this difference is smaller, but still significant both statistically and in magnitude. The result at depth 2 shows there is scope for OCT to significantly outperform CART by a large margin. We believe the smaller differences at higher depths can be attributed to the MIO problems being harder to solve, and so less progress is being made at these depths for the same time limit. Another explanation is that the natural advantage we would expect optimal methods to have over CART is less pronounced at higher depths, but this runs contrary to intuition which would suggest that larger trees have more scope for optimization than shallower ones.

Note that even at depth 1, a significant difference is present between the methods, which is simply the effect of using the misclassification score to select the best split as opposed to an impurity measure.

We have established that OCT is more likely to outperform CART, and has a small yet significant gain in out-of-sample accuracy across the collection of datasets. Next, we will consider the relative performance of the methods according to characteristics of the dataset in order to identify the types of problems where each method is more likely to outperform the other.

Table 9 presents a comparison of OCT against CART as a function of the accuracy of CART. We can see OCT consistently has an edge when the CART accuracy is above 80% or below 60%. The win rates in these areas also favor

Max. depth	CART Accuracy	CART wins	OCT wins	Ties	Accuracy Improvement
1	0–60%	1	6	3	0.11%
	60–70%	1	5	5	0.90%
	70–80%	2	6	7	0.33%
	80–90%	0	1	8	0.33%
	90–100%	0	1	7	0.01%
2	0–60%	2	5	0	3.75%
	60–70%	1	5	2	1.12%
	70–80%	3	6	4	2.87%
	80–90%	3	5	4	1.71%
	90–100%	0	4	9	0.43%
3	0–60%	3	3	0	2.53%
	60–70%	2	2	1	−0.28%
	70–80%	5	6	4	0.06%
	80–90%	3	4	4	1.22%
	90–100%	1	6	9	0.66%
4	0–60%	2	3	0	1.58%
	60–70%	4	4	1	0.11%
	70–80%	4	3	4	−0.19%
	80–90%	2	5	5	1.37%
	90–100%	1	6	9	0.60%

Table 9 Breakdown of results according to the accuracy of CART on the dataset and the maximum depth of the trees. For each instance, the number of wins for each method is shown as well as the mean accuracy improvement for OCT over CART across all datasets.

OCT. In the region where CART accuracy is 60–80%, the results are mixed with no clear winner at the higher depths. The main conclusion of this comparison is that OCT is able to consistently deliver improvements in accuracy when CART performs either well or poorly. To understand this, it seems that a high CART accuracy indicates that axis-aligned decision tree methods are well-suited to the dataset in question, which means solving the problem using MIO is more likely to lead to better decision trees with higher accuracy on the problem. Conversely, when CART performs very poorly, it may be the result of bad decisions in the trees that OCT can avoid. However, when the CART accuracy is in the middle, it could indicate that CART is hitting the limit of what is possible for a decision tree method and so the benefits of a stronger decision tree in-sample are not realized out-of-sample.

Figure 4 shows the winning method for trees of depth 2 plotted according to the accuracy of CART and the ratio of n and p. We see that the top-left corner has a very high concentration of OCT wins, and this pattern is present in both the depth 3 and 4 plots as well so they are omitted for brevity. The dashed lines on the plot indicate a region of high OCT performance, which are those datasets with CART error below 20% and n/p over 10. This region was determined by looking at the results collectively across all depths.

Table 10 summarizes the results for each depth according to whether they are in this region of high performance. We see that in the regions of high performance, OCT is almost always the winning method if there is a winner, and the average

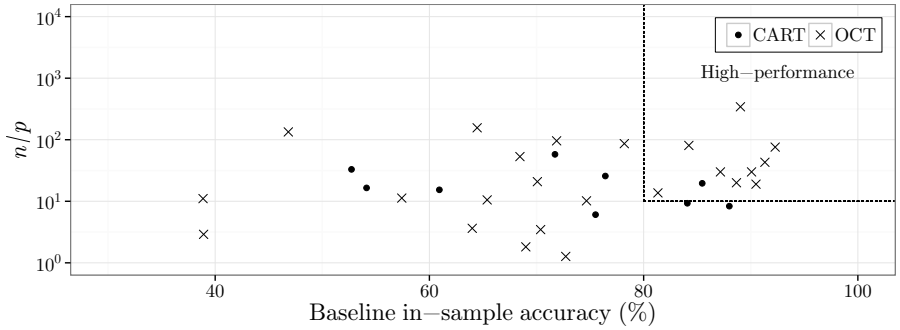


Fig. 4 Plot of winning method (CART vs. OCT) by the accuracy of CART and the ratio of the number of points to the dimension of points, n/p, in each dataset for trees of depth 2. The high-performance where OCT is highly likely to perform strongest is indicated by the dashed lines.

Max. depth	In High-perf. Region?	CART wins	OCT wins	Ties	Accuracy Improvement
1	✓	0	1	13	0.01%
	✗	4	18	17	0.49%
2	✓	1	9	12	1.31%
	✗	8	16	7	2.25%
3	✓	2	9	12	1.15%
	✗	12	12	6	0.41%
4	✓	1	10	13	1.19%
	✗	12	11	6	0.15%

Table 10 Breakdown of results according to whether the dataset is in the region of high-performance and the maximum depth of the trees. For each instance, the number of wins for each method is shown as well as the mean accuracy improvement for OCT over CART across all datasets.

improvement for OCT is consistently around 1.2–1.3%. For the datasets that are not in this region, the results are more mixed and there is no clear winner, although at depth 2 there is a large difference of 2.2% between the accuracies of CART and OCT, and in all other cases OCT has a slight accuracy advantage.

This breakdown suggests that we can predict with high probability that OCT is very likely to outperform CART if the following expression is true:

$$(\text{CART accuracy} \geq 80\%) \wedge (n/p \geq 10)$$

This rule makes sense intuitively, as datasets where CART performs well are likely those which are well suited to decision tree models, which means that creating these trees optimally could well lead to better performance, whereas this may not be as likely if tree models are a poor fit to the data. Additionally, if $n \gg p$, this suggests we have enough data relative to the number of possible splits to learn

Max. depth	CART wins	OCT-H wins	Ties	Accuracy improvement	p-value
1	3	36	14	5.12%	$\sim 10^{-16}$
2	10	32	11	4.88%	$\sim 10^{-14}$
3	13	31	9	3.59%	$\sim 10^{-12}$
4	13	29	11	3.12%	$\sim 10^{-11}$

Table 11 Comparison of CART and OCT-H across a range of depths, showing the number of datasets for which each method had the highest out-of-sample accuracy, and the mean improvement in out-of-sample accuracy when using OCT-H across all datasets along with the p-value indicating the statistical significance of this difference.

a good tree without overfitting, echoing the results of the second synthetic test in Section 5.

The other key takeaway is that if we are not in this region, there is no indication of any ill-effects; the results outside this region are balanced, and there is even a slight edge towards Optimal Trees.

Based on the strength of these results, this partitioning based on n/p and CART accuracy should provide a highly accurate and useful guide for when OCT are likely to yield accuracy improvements on real-world problems.

6.3 Optimal Classification Trees with Hyperplanes vs. CART

We now present a comparison of out-of-sample accuracies between OCT-H and CART. Recall that after formulating the OCT problem using MIO, we were able to easily obtain the OCT-H problem by relaxing integrality constraints on the split variables $\mathbf{a}$. This comparison aims to examine the possible accuracy gains from first formulating the problem using MIO and then relaxing a subset of difficult constraints to generate a model that is simpler to solve.

The entire set of mean out-of-sample accuracies on each dataset for both methods at depth 2 is provided in Table 12, which again reports the size and dimension of the dataset, the number of classes K, and the average out-of-sample accuracy of each method along with the mean improvement of OCT-H over CART and the associated standard error (results for depths 1, 3 and 4 are provided in Appendix B).

Table 11 shows a summary of the relative performance of CART and OCT-H according to the depth of the trees. We see that at all depths, OCT is stronger than CART on the vast majority of datasets. OCT-H also gives a significant increase of 3–5% in out-of-sample accuracy compared to CART across all datasets, depending on the tree depth. The corresponding p-values confirm this difference to be highly significant statistically. We believe the smaller differences at higher depths can be attributed to the MIO problems being harder to solve, and so less progress is being made at these depths for the same time limit. Similar to the OCT comparison, the advantage of OCT-H over CART decreases as the depth of the trees increases. This could be due to the increasing difficulty of the problems as the depth increases as for OCT, but it may also be the case that most of the accuracy advantage from multivariate decision trees can be realized with shallower trees, and so the

Dataset				Mean out-of-sample accuracy		Mean improvement
Name	n	p	K	CART	OCT-H	
acute-inflammations-1	120	6	2	88.7	**100.0**	$+11.33 \pm 1.70$
acute-inflammations-2	120	6	2	**100.0**	**100.0**	0.00 ± 0.00
balance-scale	625	4	3	64.5	**87.6**	$+23.18 \pm 1.61$
banknote-authentication	1372	4	2	89.0	**91.5**	$+2.51 \pm 6.86$
blood-transfusion	748	4	2	75.5	**77.2**	$+1.71 \pm 0.62$
breast-cancer-diagnostic	569	30	2	90.5	**93.1**	$+2.66 \pm 1.16$
breast-cancer-prognostic	194	32	2	**75.5**	**75.5**	0.00 ± 0.00
breast-cancer	683	9	2	92.3	**97.0**	$+4.68 \pm 0.76$
car-evaluation	1728	15	4	73.7	**87.5**	$+13.80 \pm 0.69$
chess-king-rook-vs-king-pawn	3196	37	2	78.2	**94.9**	$+16.75 \pm 1.57$
climate-model-crashes	540	18	2	90.1	**92.9**	$+2.81 \pm 0.64$
congressional-voting-records	232	16	2	**98.6**	**98.6**	0.00 ± 0.00
connectionist-bench-sonar	208	60	2	**70.4**	70.0	-0.38 ± 1.12
connectionist-bench	990	10	11	24.8	**24.9**	$+0.08 \pm 0.08$
contraceptive-method-choice	1473	11	3	**46.8**	**46.8**	0.00 ± 0.00
credit-approval	653	37	2	87.7	**87.9**	$+0.12 \pm 0.12$
cylinder-bands	277	484	2	62.6	**66.4**	$+3.77 \pm 1.98$
dermatology	358	34	6	65.4	**74.2**	$+8.76 \pm 2.81$
echocardiogram	61	6	2	74.7	**77.3**	$+2.67 \pm 1.63$
fertility	100	12	2	**88.0**	**88.0**	0.00 ± 0.00
haberman-survival	306	3	2	**73.2**	73.0	-0.26 ± 0.26
hayes-roth	132	4	3	52.7	**61.2**	$+8.48 \pm 5.70$
heart-disease-cleveland	297	18	5	54.1	**54.7**	$+0.53 \pm 0.90$
hepatitis	80	19	2	**83.0**	81.0	-2.00 ± 1.22
image-segmentation	210	19	7	38.9	**49.1**	$+10.19 \pm 2.96$
indian-liver-patient	579	10	2	71.7	**72.6**	$+0.83 \pm 0.67$
ionosphere	351	34	2	**87.8**	86.2	-1.61 ± 2.08
iris	150	4	3	92.4	**95.1**	$+2.70 \pm 1.71$
mammographic-mass	830	10	2	**81.2**	**81.2**	0.00 ± 0.00
monks-problems-1	124	11	2	57.4	**93.5**	$+36.13 \pm 4.00$
monks-problems-2	169	11	2	60.9	**75.8**	$+14.88 \pm 6.18$
monks-problems-3	122	11	2	**94.2**	92.3	-1.94 ± 1.94
optical-recognition	3823	64	10	**29.7**	29.3	-0.38 ± 0.38
ozone-level-detection-eight	1847	72	2	**93.1**	**93.1**	0.00 ± 0.00
ozone-level-detection-one	1848	72	2	**96.8**	**96.8**	0.00 ± 0.00
parkinsons	195	21	2	84.1	**84.9**	$+0.82 \pm 1.04$
pima-indians-diabetes	768	8	2	**71.9**	71.4	-0.52 ± 1.42
planning-relax	182	12	2	**71.1**	70.7	-0.44 ± 0.44
qsar-biodegradation	1055	41	2	76.4	**83.0**	$+6.54 \pm 1.67$
seeds	210	7	3	87.2	**90.6**	$+3.40 \pm 1.83$
seismic-bumps	2584	20	2	**93.3**	**93.3**	0.00 ± 0.00
soybean-small	47	37	4	72.7	**98.2**	$+25.45 \pm 1.82$
spambase	4601	57	2	84.2	**85.7**	$+1.46 \pm 1.44$
spect-heart	80	22	2	64.0	**70.0**	$+6.00 \pm 2.92$
spectf-heart	80	44	2	**69.0**	67.0	-2.00 ± 4.90
statlog-project-german-credit	1000	48	2	70.1	**70.4**	$+0.32 \pm 0.32$
statlog-project-landsat-satellite	4435	36	6	**63.2**	**63.2**	0.00 ± 0.00
teaching-assistant-evaluation	151	52	3	38.9	**55.7**	$+16.76 \pm 4.63$
thoracic-surgery	470	24	2	**85.5**	83.9	-1.54 ± 1.54
thyroid-disease-ann-thyroid	3772	21	3	**95.6**	92.5	-3.10 ± 0.06
thyroid-disease-new-thyroid	215	5	3	91.3	**95.8**	$+4.53 \pm 0.75$
tic-tac-toe-endgame	958	18	2	68.5	**97.0**	$+28.54 \pm 0.41$
wall-following-robot-2	5456	2	4	**94.0**	**94.0**	0.00 ± 0.00
wine	178	13	3	81.3	**91.1**	$+9.78 \pm 3.76$

Table 12 Full results for CART and OCT-H at depth 2.

Max. depth	CART Accuracy	CART wins	OCT-H wins	Ties	Accuracy Improvement
1	0–60%	0	7	3	6.74%
	60–70%	0	10	1	8.18%
	70–80%	1	11	3	6.00%
	80–90%	1	5	3	1.73%
	90–100%	1	3	4	1.05%
2	0–60%	0	6	1	10.31%
	60–70%	1	6	1	10.39%
	70–80%	4	8	1	5.11%
	80–90%	3	7	2	2.02%
	90–100%	2	5	6	0.95%
3	0–60%	1	5	0	6.13%
	60–70%	1	4	0	9.12%
	70–80%	3	9	3	4.38%
	80–90%	5	6	0	2.06%
	90–100%	3	7	6	1.21%
4	0–60%	1	3	1	4.96%
	60–70%	1	8	0	5.17%
	70–80%	3	5	3	4.66%
	80–90%	5	6	1	1.95%
	90–100%	3	7	6	1.22%

Table 13 Breakdown of results according to the accuracy of CART on the dataset and the maximum depth of the trees. For each instance, the number of wins for each method is shown as well as the mean accuracy improvement for OCT-H over CART across all datasets.

advantages of deeper trees are less pronounced, allowing CART to reduce the accuracy gap with increasing depth.

Unlike the OCT vs. CART comparison, the depth 1 difference here is very significant because multivariate trees of depth 1 can still use multiple variables in the splits, and this leads to a very significant mean improvement of around 5%.

As we did for OCT, we will now consider the influence of problem characteristics on the performance of the OCT-H.

Table 13 compares OCT-H to CART broken down by the accuracy of CART. Unlike OCT, we see that OCT-H improves significantly upon CART accuracy in each group, with the most significant gains coming from the datasets where CART is weakest (below 80%), showing average improvements of 5–10%. As the CART accuracy increases, the improvement offered by OCT-H decreases, but they still have an advantage even at the highest CART accuracies. This indicates that multivariate trees are able to exploit problem structure in a way that univariate trees cannot, and this exploitation is most beneficial when univariate decision trees perform poorly.

Figure 5 shows the winning method for each dataset as a function of the CART accuracy on each dataset and the number of features in the data. As we did for OCT, we can identify a region where OCT-H is highly likely to outperform CART. Like OCT, we have considered the results for all depths when identifying this region but the plots for other depths show similar patterns and are omitted. We see that this region of high-performance is when either the CART accuracy is low or the

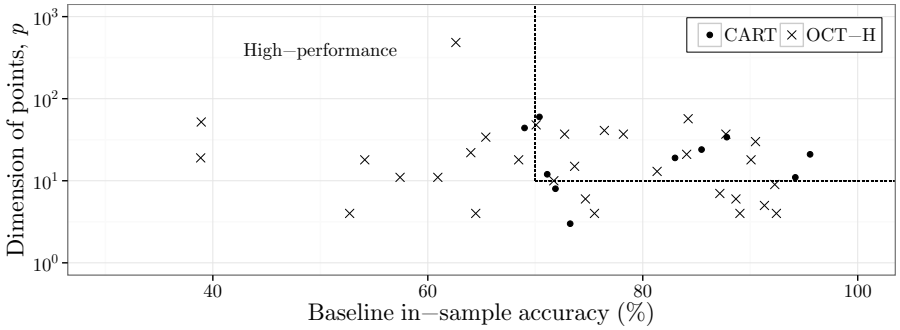


Fig. 5 Plot of winning method (CART vs. OCT-H) by the accuracy of CART and the dimension of points, p, in each dataset for trees of depth 2. The high-performance where OCT-H is highly likely to perform strongest is indicated by the dashed lines.

Max. depth	In High-perf. Region?	CART wins	OCT-H wins	Ties	Accuracy Improvement
1	✓	0	27	6	6.23%
	✗	3	9	8	3.29%
2	✓	3	21	5	6.51%
	✗	7	11	6	2.90%
3	✓	5	19	2	5.21%
	✗	8	12	7	2.02%
4	✓	5	19	4	4.25%
	✗	8	10	7	1.86%

Table 14 Breakdown of results according to whether the dataset is in the region of high-performance and the maximum depth of the trees. For each instance, the number of wins for each method is shown as well as the mean accuracy improvement for OCT-H over CART across all datasets.

number of features is low, with the following test determining membership of the region:

$$(\text{CART Accuracy} \leq 70\%) \vee (p \leq 10)$$

Table 14 shows the relative performance of the methods in each of the regions. We see that in terms of the win rate, OCT-H is highly likely to deliver significant out-of-sample accuracy improvements on datasets that fall into the high-performance region, and in this region the average accuracy improvement over CART is 4–7% depending on depth. Outside the region, OCT-H is still likely to beat CART, but less so than in the region, and the average accuracy improvment is 2–3%. We have already addressed why OCT-H is likely to do well when the CART accuracy is poor, but these results also suggest that OCT-H performs better when the dimension of the data is small. We presume that this may be because there are fewer possible hyperplane splits to consider and so the feasible space for the problem is smaller, allowing better solutions to be found faster.

6.4 Optimal Classification Trees vs. Random Forests

We conclude our experiments on real-world datasets by providing a brief comparison with Random Forests (Breiman 2001). Random forests achieve state-of-the-art accuracies, and so this allows us to place our results in a wider context. We would like to note that our goal in developing these methods was not necessarily to develop the best classification method overall; rather, we intended to show the benefits of taking a problem that is traditionally solved by a heuristic and instead solving this problem to optimality. Nevertheless, these comparisons help to quantify the significance of the improvements in accuracy.

We ran random forests on all datasets using the RANDOMFOREST package (Liaw and Wiener 2002) in R, using the same training, validation and testing splits as the other methods. The random forests do not require any tuning, so they were trained on the combined training and validation sets and then evaluated on the testing set. This was averaged over five splits as for the other methods. The forests were trained with 100 trees.

The average accuracy of Random Forests across all 53 datasets was 85.8%, which is about 6%, 5% and 3% higher than CART, OCT and OCT-H at depth 4, respectively. We see that across all datasets, OCT closes the gap between CART and Random Forests by about one-sixth, and OCT-H by about half. We believe this demonstrates the significance of our accuracy improvements when measured against the improvement offered by Random Forests.

We found that Random Forests performed strongest relative to OCT-H on datasets with a high number of classes, K, or features, p. If we consider only datasets with $K = 2$ or $K = 3$ classes, and $p \leq 25$, we find that OCT-H and Random Forests are comparable in accuracy, with average out-of-sample accuracies of 85.2% and 85.1%, respectively. We can therefore achieve state-of-the-art accuracies in this regime with just a single tree learner, allowing us to maintain some interpretability in the model, albeit less than an axis-aligned tree, but still much more than a forest. We note that 31 of the datasets we considered fall into this regime, so we do not believe it is a restrictive setting.

Finally, we note that we did not tune the value of $N_{\min}$ in our experiments, nor did we consider other improvements to the original OCT problem that could improve accuracy. With the right selection of model and parameters, we believe it is possible to train a single decision tree learner that has the accuracy of a random forest without sacrificing interpretability, which would be significant for those applications where interpretability is required.

7 Conclusions

In this paper, we have revisited the classical problem of decision tree creation under a modern MIO lens. We presented a novel MIO formulation for creating optimal decision trees that captures the discrete nature of this problem and motivates two new methods, *Optimal Classification Trees* (OCT) and *Optimal Classification Trees with Hyperplanes* (OCT-H).

Experiments with synthetic data provide strong evidence that optimal decision trees can better recover the true generating decision tree in the data, contrary to

the popular belief that such optimal methods will just overfit the training data at the expense of generalization ability.

Exploiting the astonishing progress of MIO solvers in the last twenty-five years, our comprehensive computational experiments showed that both the OCT and OCT-H problems are both practically solvable and deliver solutions that outperform the classical state-of-the-art methods, with average absolute improvements in out-of-sample accuracy over CART of 1–2% for OCT and 3–5% for OCT-H across all datasets, depending on the depth of the tree. We also provided guidance for each method that predict when each is most likely to outperform CART.

For OCT, we predict consistent accuracy improvements of 2–4% when the CART accuracy is below 60%, and improvements of 1.2–1.3% when the CART accuracy is above 80% and there is sufficient training data available relative to the complexity of the problem ($n/p \geq 10$). Outside these cases, we find that OCT and CART are competitive with each other.

For OCT-H, we can predict consistent accuracy improvements of 4–7% over CART if the CART accuracy is below 70% or the dimension of the data is no more than 10. On other datasets, we can still predict strong performance by OCT-H, but with a smaller yet still significant average improvement of 2–3%.

These results provide comprehensive evidence that the optimal decision tree problem is tractable for practical applications and leads to significant improvements over heuristic methods.

Acknowledgements

We thank the three reviewers and the action editor for their comments that improved the paper substantially. We also thank Prof. Andrew Mason for comments on our earlier MIO formulation, and Jenny McLean for editorial comments that improved the exposition of the paper.

References

Arthanari T, Dodge Y (1981) *Mathematical Programming in Statistics*, vol 341. Wiley, New York

Auer P, Holte RC, Maass W (1995) Theory and applications of agnostic pac-learning with small decision trees. In: Proceedings of the Twelfth International Conference on Machine Learning, pp 21–29

Bennett KP (1992) Decision tree construction via linear programming. In: Evans M (ed) Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society Conference, pp 97–101

Bennett KP, Blue J (1996) Optimal decision trees. *Rensselaer Polytechnic Institute Math Report* 214

Bennett KP, Blue JA (1998) A support vector machine approach to decision trees. In: Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on, IEEE, vol 3, pp 2396–2401

Bertsimas D, King A (2015a) An algorithmic approach to linear regression. *Operations Research*

Bertsimas D, King A (2015b) Logistic regression: From art to science. *Operations Research Center, MIT, submitted for publication*
Bertsimas D, Mazumder R (2014) Least quantile regression via modern optimization. *The Annals of Statistics* 42(6):2494–2525
Bertsimas D, Shioda R (2007) Classification and regression via integer optimization. *Operations Research* 55(2):252–271
Bertsimas D, Weismantel R (2005) *Optimization over Integers*. Dynamic Ideas, Belmont, Massachusetts
Bertsimas D, King A, Mazumder R (2015) Best subset selection via a modern optimization lens. *Annals of Statistics, to appear*
Bezanson J, Edelman A, Karpinski S, Shah VB (2014) Julia: A fresh approach to numerical computing. *arXiv preprint arXiv:14111607*
Bixby RE (2012) A brief history of linear and mixed-integer programming computation. *Documenta Mathematica* pp 107–121
Breiman L (2001) Random forests. *Machine Learning* 45(1):5–32
Breiman L, Friedman J, Olshen R, Stone C (1984) *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, California
Cox Jr LA, Yuping Q, Kuehner W (1989) Heuristic least-cost computation of discrete classification functions with uncertain argument values. *Annals of Operations Research* 21(1):1–29
Esmeir S, Markovitch S (2007) Anytime learning of decision trees. *The Journal of Machine Learning Research* 8:891–933
Gurobi Optimization Inc (2015a) Gurobi 6.0 performance benchmarks. http://www.gurobi.com/pdfs/benchmarks.pdf, accessed 5 September 2015
Gurobi Optimization Inc (2015b) Gurobi optimizer reference manual. http://www.gurobi.com
Heath D, Kasif S, Salzberg S (1993) Induction of oblique decision trees. In: IJCAI, Citeseer, pp 1002–1007
Hyafil L, Rivest RL (1976) Constructing optimal binary decision trees is np-complete. *Information Processing Letters* 5(1):15–17
IBM ILOG CPLEX (2014) V12.1 users manual
Liaw A, Wiener M (2002) Classification and regression by randomforest. *R News* 2(3):18–22, URL http://CRAN.R-project.org/doc/Rnews/
Lichman M (2013) UCI machine learning repository. URL http://archive.ics.uci.edu/ml
Loh WY, Shih YS (1997) Split selection methods for classification trees. *Statistica sinica* pp 815–840
López-Chau A, Cervantes J, López-García L, Lamont FG (2013) Fishers decision tree. *Expert Systems with Applications* 40(16):6283–6291
Lubin M, Dunning I (2015) Computing in operations research using julia. *INFORMS Journal on Computing* 27(2):238–248
Murthy S, Salzberg S (1995a) Lookahead and pathology in decision tree induction. In: IJCAI, Citeseer, pp 1025–1033
Murthy SK, Salzberg S (1995b) Decision tree induction: How effective is the greedy heuristic? In: KDD, pp 222–227
Murthy SK, Kasif S, Salzberg S (1994) A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research* 2:1–32
Nemhauser GL (2013) Integer Programming: the Global Impact. Presented at EURO, INFORMS, Rome, Italy, 2013. http://euro-informs2013.org/data/http_

/euro2013.org/wp-content/uploads/nemhauser.pdf, accessed 9 September 2015
Norouzi M, Collins M, Johnson MA, Fleet DJ, Kohli P (2015) Efficient non-greedy optimization of decision trees. In: Advances in Neural Information Processing Systems, pp 1720–1728
Norton SW (1989) Generating better decision trees. In: IJCAI, vol 89, pp 800–805
Payne HJ, Meisel WS (1977) An algorithm for constructing optimal binary decision trees. *Computers, IEEE Transactions on* 100(9):905–916
Quinlan JR (1986) Induction of decision trees. *Machine Learning* 1(1):81–106
Quinlan JR (1993) *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA
R Core Team (2015) *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, URL http://www.R-project.org/
Son NH (1998) From optimal hyperplanes to optimal decision trees. *Fundamenta Informaticae* 34(1, 2):145–174
Therneau T, Atkinson B, Ripley B (2015) *rpart: Recursive Partitioning and Regression Trees*. URL http://CRAN.R-project.org/package=rpart, R package version 4.1-9
Tjortjis C, Keane J (2002) T3: a classification algorithm for data mining. In: Lecture Notes in Computer Science, Springer, vol 2412, pp 50–55
Top500 Supercomputer Sites (2015) Performance development. http://www.top500.org/statistics/perfdevel/, accessed 4 September 2015
Truong A (2009) Fast growing and interpretable oblique trees via logistic regression models. PhD thesis, University of Oxford
Tzirakis P, Tjortjis C (2016) T3c: improving a decision tree classification algorithms interval splits on continuous attributes. *Advances in Data Analysis and Classification* pp 1–18
Wickramarachchi D, Robertson B, Reale M, Price C, Brown J (2016) Hhcart: An oblique decision tree. *Computational Statistics & Data Analysis* 96:12–23

A Additional Results for OCT vs. CART

Dataset				Mean out-of-sample accuracy		Mean improvement
Name	n	p	K	CART	OCT	
acute-inflammations-1	120	6	2	95.3	**100.0**	$+4.67 \pm 2.91$
acute-inflammations-2	120	6	2	**100.0**	**100.0**	0.00 ± 0.00
balance-scale	625	4	3	**70.4**	68.9	-1.53 ± 0.87
banknote-authentication	1372	4	2	89.0	**89.6**	$+0.58 \pm 0.58$
blood-transfusion	748	4	2	**77.5**	77.0	-0.53 ± 0.41
breast-cancer-diagnostic	569	30	2	90.5	**91.5**	$+0.98 \pm 0.82$
breast-cancer-prognostic	194	32	2	**75.5**	**75.5**	0.00 ± 0.00
breast-cancer	683	9	2	93.1	**95.3**	$+2.22 \pm 1.00$
car-evaluation	1728	15	4	**77.4**	**77.4**	0.00 ± 0.00
chess-king-rook-vs-king-pawn	3196	37	2	**90.6**	**90.6**	0.00 ± 0.00
climate-model-crashes	540	18	2	91.0	**91.4**	$+0.44 \pm 0.30$
congressional-voting-records	232	16	2	**98.6**	**98.6**	0.00 ± 0.00
connectionist-bench-sonar	208	60	2	**70.8**	69.6	-1.15 ± 1.88
connectionist-bench	990	10	11	34.4	**36.4**	$+2.02 \pm 0.44$
contraceptive-method-choice	1473	11	3	**50.8**	50.1	-0.70 ± 1.87
credit-approval	653	37	2	**87.7**	**87.7**	0.00 ± 0.00
cylinder-bands	277	484	2	**61.7**	**61.7**	0.00 ± 0.00
dermatology	358	34	6	**78.7**	**78.7**	0.00 ± 0.00
echocardiogram	61	6	2	72.0	**74.7**	$+2.67 \pm 2.67$
fertility	100	12	2	86.4	**86.4**	0.00 ± 1.26
haberman-survival	306	3	2	**73.5**	73.2	-0.26 ± 0.26
hayes-roth	132	4	3	**56.4**	53.3	-3.03 ± 1.66
heart-disease-cleveland	297	18	5	**54.9**	54.7	-0.27 ± 0.27
hepatitis	80	19	2	**82.0**	81.0	-1.00 ± 1.00
image-segmentation	210	19	7	52.5	**65.3**	$+12.83 \pm 1.10$
indian-liver-patient	579	10	2	71.6	**71.7**	$+0.14 \pm 0.14$
ionosphere	351	34	2	**87.8**	87.6	-0.23 ± 0.23
iris	150	4	3	92.4	**93.5**	$+1.08 \pm 1.08$
mammographic-mass	830	10	2	**81.7**	**81.7**	0.00 ± 0.00
monks-problems-1	124	11	2	65.8	**70.3**	$+4.52 \pm 4.63$
monks-problems-2	169	11	2	**60.9**	60.0	-0.93 ± 0.93
monks-problems-3	122	11	2	**94.2**	**94.2**	0.00 ± 0.00
optical-recognition	3823	64	10	41.4	**41.6**	$+0.21 \pm 0.21$
ozone-level-detection-eight	1847	72	2	**93.1**	**93.1**	0.00 ± 0.00
ozone-level-detection-one	1848	72	2	**96.8**	**96.8**	0.00 ± 0.00
parkinsons	195	21	2	86.1	**86.5**	$+0.41 \pm 0.41$
pima-indians-diabetes	768	8	2	70.6	**71.1**	$+0.52 \pm 1.61$
planning-relax	182	12	2	**71.1**	**71.1**	0.00 ± 0.00
qsar-biodegradation	1055	41	2	78.5	**78.6**	$+0.15 \pm 0.09$
seeds	210	7	3	86.8	**87.9**	$+1.13 \pm 1.28$
seismic-bumps	2584	20	2	**93.3**	**93.3**	0.00 ± 0.00
soybean-small	47	37	4	**100.0**	98.2	-1.82 ± 1.82
spambase	4601	57	2	**86.0**	**86.0**	0.00 ± 0.00
spect-heart	80	22	2	64.0	**67.0**	$+3.00 \pm 3.00$
spectf-heart	80	44	2	**69.0**	61.0	-8.00 ± 6.24
statlog-project-german-credit	1000	48	2	**70.7**	70.5	-0.24 ± 0.68
statlog-project-landsat-satellite	4435	36	6	77.7	**77.9**	$+0.18 \pm 0.08$
teaching-assistant-evaluation	151	52	3	38.9	**43.2**	$+4.32 \pm 2.02$
thoracic-surgery	470	24	2	**85.5**	84.6	-0.85 ± 0.85
thyroid-disease-ann-thyroid	3772	21	3	**95.6**	**95.6**	0.00 ± 0.00
thyroid-disease-new-thyroid	215	5	3	91.3	**94.3**	$+3.02 \pm 1.28$
tic-tac-toe-endgame	958	18	2	73.1	**74.1**	$+1.00 \pm 0.54$
wall-following-robot-2	5456	2	4	**100.0**	**100.0**	0.00 ± 0.00
wine	178	13	3	80.9	**94.2**	$+13.33 \pm 1.86$

Table 15 Full results for CART and OCT at depth 3.

Dataset				Mean out-of-sample accuracy		Mean improvement
Name	n	p	K	CART	OCT	
acute-inflammations-1	120	6	2	95.3	**100.0**	$+4.67 \pm 2.91$
acute-inflammations-2	120	6	2	**100.0**	**100.0**	0.00 ± 0.00
balance-scale	625	4	3	**73.4**	71.6	-1.78 ± 1.05
banknote-authentication	1372	4	2	89.0	**90.7**	$+1.63 \pm 1.06$
blood-transfusion	748	4	2	**77.5**	77.0	-0.53 ± 0.41
breast-cancer-diagnostic	569	30	2	90.5	**91.5**	$+0.98 \pm 0.82$
breast-cancer-prognostic	194	32	2	**75.5**	**75.5**	0.00 ± 0.00
breast-cancer	683	9	2	93.1	**95.3**	$+2.22 \pm 1.00$
car-evaluation	1728	15	4	**78.8**	**78.8**	0.00 ± 0.00
chess-king-rook-vs-king-pawn	3196	37	2	**90.4**	**90.4**	0.00 ± 0.00
climate-model-crashes	540	18	2	91.0	**91.1**	$+0.15 \pm 0.15$
congressional-voting-records	232	16	2	**98.6**	**98.6**	0.00 ± 0.00
connectionist-bench-sonar	208	60	2	**70.0**	69.6	-0.38 ± 2.68
connectionist-bench	990	10	11	43.9	**44.6**	$+0.73 \pm 0.65$
contraceptive-method-choice	1473	11	3	**53.8**	53.3	-0.49 ± 0.92
credit-approval	653	37	2	**87.5**	**87.5**	0.00 ± 0.00
cylinder-bands	277	484	2	**62.6**	**62.6**	0.00 ± 0.00
dermatology	358	34	6	**89.2**	**89.2**	0.00 ± 0.00
echocardiogram	61	6	2	**72.0**	**72.0**	0.00 ± 0.00
fertility	100	12	2	**88.0**	**88.0**	0.00 ± 0.00
haberman-survival	306	3	2	72.7	**73.2**	$+0.52 \pm 0.52$
hayes-roth	132	4	3	56.4	**59.4**	$+3.03 \pm 2.54$
heart-disease-cleveland	297	18	5	**54.9**	53.6	-1.33 ± 1.33
hepatitis	80	19	2	**82.0**	81.0	-1.00 ± 1.00
image-segmentation	210	19	7	64.2	**72.8**	$+8.68 \pm 2.90$
indian-liver-patient	579	10	2	69.8	**71.7**	$+1.93 \pm 1.76$
ionosphere	351	34	2	**87.8**	87.6	-0.23 ± 0.23
iris	150	4	3	92.4	**93.5**	$+1.08 \pm 1.08$
mammographic-mass	830	10	2	**81.7**	**81.7**	0.00 ± 0.00
monks-problems-1	124	11	2	68.4	**74.2**	$+5.81 \pm 3.87$
monks-problems-2	169	11	2	**62.8**	54.0	-8.84 ± 3.99
monks-problems-3	122	11	2	**94.2**	**94.2**	0.00 ± 0.00
optical-recognition	3823	64	10	**54.7**	54.7	-0.02 ± 0.02
ozone-level-detection-eight	1847	72	2	**93.1**	**93.1**	0.00 ± 0.00
ozone-level-detection-one	1848	72	2	**96.8**	**96.8**	0.00 ± 0.00
parkinsons	195	21	2	86.1	**86.5**	$+0.41 \pm 0.41$
pima-indians-diabetes	768	8	2	71.7	**72.4**	$+0.73 \pm 0.45$
planning-relax	182	12	2	**71.1**	**71.1**	0.00 ± 0.00
qsar-biodegradation	1055	41	2	79.6	**79.8**	$+0.15 \pm 0.09$
seeds	210	7	3	86.8	**89.1**	$+2.26 \pm 1.51$
seismic-bumps	2584	20	2	**93.3**	**93.3**	0.00 ± 0.00
soybean-small	47	37	4	**100.0**	98.2	-1.82 ± 1.82
spambase	4601	57	2	86.0	**86.1**	$+0.09 \pm 0.09$
spect-heart	80	22	2	64.0	**66.0**	$+2.00 \pm 2.00$
spectf-heart	80	44	2	**70.0**	62.0	-8.00 ± 5.61
statlog-project-german-credit	1000	48	2	**69.9**	69.8	-0.16 ± 0.16
statlog-project-landsat-satellite	4435	36	6	**78.2**	78.0	-0.25 ± 0.28
teaching-assistant-evaluation	151	52	3	45.4	**51.4**	$+5.95 \pm 5.01$
thoracic-surgery	470	24	2	**85.5**	**85.5**	0.00 ± 0.00
thyroid-disease-ann-thyroid	3772	21	3	**95.6**	**95.6**	0.00 ± 0.00
thyroid-disease-new-thyroid	215	5	3	91.3	**93.6**	$+2.26 \pm 1.10$
tic-tac-toe-endgame	958	18	2	**74.2**	73.3	-0.92 ± 0.92
wall-following-robot-2	5456	2	4	**100.0**	**100.0**	0.00 ± 0.00
wine	178	13	3	80.9	**94.2**	$+13.33 \pm 1.86$

Table 16 Full results for CART and OCT at depth 4.

B Additional Results for OCT-H vs. CART

Dataset				Mean out-of-sample accuracy		Mean improvement
Name	n	p	K	CART	OCT-H	
acute-inflammations-1	120	6	2	78.7	**100.0**	+21.33 ± 3.09
acute-inflammations-2	120	6	2	92.0	**97.3**	+5.33 ± 1.70
balance-scale	625	4	3	60.9	**87.6**	+26.75 ± 0.73
banknote-authentication	1372	4	2	83.6	**89.8**	+6.18 ± 8.63
blood-transfusion	748	4	2	75.9	**77.2**	+1.28 ± 0.69
breast-cancer-diagnostic	569	30	2	88.5	**93.1**	+4.62 ± 1.39
breast-cancer-prognostic	194	32	2	**75.5**	**75.5**	0.00 ± 0.00
breast-cancer	683	9	2	92.2	**97.0**	+4.80 ± 0.73
car-evaluation	1728	15	4	69.9	**87.5**	+17.55 ± 0.35
chess-king-rook-vs-king-pawn	3196	37	2	66.8	**94.9**	+28.14 ± 1.41
climate-model-crashes	540	18	2	91.9	**93.2**	+1.33 ± 0.82
congressional-voting-records	232	16	2	**98.6**	**98.6**	0.00 ± 0.00
connectionist-bench-sonar	208	60	2	**70.4**	**70.4**	0.00 ± 1.49
connectionist-bench	990	10	11	**16.2**	**16.2**	0.00 ± 0.00
contraceptive-method-choice	1473	11	3	42.8	**45.4**	+2.55 ± 1.66
credit-approval	653	37	2	87.7	**87.9**	+0.12 ± 0.12
cylinder-bands	277	484	2	63.8	**65.5**	+1.74 ± 1.41
dermatology	358	34	6	49.4	**50.3**	+0.90 ± 0.42
echocardiogram	61	6	2	72.0	**76.0**	+4.00 ± 1.63
fertility	100	12	2	**88.0**	**88.0**	0.00 ± 0.00
haberman-survival	306	3	2	72.7	**73.0**	+0.26 ± 0.26
hayes-roth	132	4	3	44.8	**60.6**	+15.76 ± 3.37
heart-disease-cleveland	297	18	5	50.7	**53.9**	+3.20 ± 1.55
hepatitis	80	19	2	**83.0**	**83.0**	0.00 ± 0.00
image-segmentation	210	19	7	**26.4**	**26.4**	0.00 ± 0.00
indian-liver-patient	579	10	2	71.7	**72.3**	+0.55 ± 0.40
ionosphere	351	34	2	80.7	**85.3**	+4.60 ± 3.49
iris	150	4	3	**64.9**	**64.9**	0.00 ± 0.00
mammographic-mass	830	10	2	**81.2**	**81.2**	0.00 ± 0.00
monks-problems-1	124	11	2	50.3	**83.9**	+33.55 ± 2.99
monks-problems-2	169	11	2	60.5	**65.1**	+4.65 ± 2.94
monks-problems-3	122	11	2	74.2	**92.3**	+18.06 ± 4.16
optical-recognition	3823	64	10	**19.4**	**19.4**	0.00 ± 0.00
ozone-level-detection-eight	1847	72	2	**93.1**	**93.1**	0.00 ± 0.00
ozone-level-detection-one	1848	72	2	**96.8**	**96.8**	0.00 ± 0.00
parkinsons	195	21	2	84.9	**86.5**	+1.63 ± 1.76
pima-indians-diabetes	768	8	2	70.8	**71.6**	+0.73 ± 0.73
planning-relax	182	12	2	**71.1**	69.8	−1.33 ± 1.33
qsar-biodegradation	1055	41	2	73.7	**83.3**	+9.58 ± 1.85
seeds	210	7	3	64.5	**64.9**	+0.38 ± 1.10
seismic-bumps	2584	20	2	**93.3**	**93.3**	0.00 ± 0.00
soybean-small	47	37	4	**54.5**	**54.5**	0.00 ± 0.00
spambase	4601	57	2	78.4	**83.6**	+5.16 ± 1.02
spect-heart	80	22	2	64.0	**70.0**	+6.00 ± 2.92
spectf-heart	80	44	2	68.0	**69.0**	+1.00 ± 3.32
statlog-project-german-credit	1000	48	2	70.0	**71.6**	+1.60 ± 1.17
statlog-project-landsat-satellite	4435	36	6	43.9	**44.0**	+0.14 ± 0.14
teaching-assistant-evaluation	151	52	3	35.7	**47.0**	+11.35 ± 3.67
thoracic-surgery	470	24	2	**85.5**	83.9	−1.54 ± 1.54
thyroid-disease-ann-thyroid	3772	21	3	**95.6**	92.5	−3.10 ± 0.06
thyroid-disease-new-thyroid	215	5	3	78.5	**82.3**	+3.77 ± 1.33
tic-tac-toe-endgame	958	18	2	70.6	**97.2**	+26.61 ± 1.13
wall-following-robot-2	5456	2	4	**78.8**	**78.8**	0.00 ± 0.00
wine	178	13	3	64.0	**66.2**	+2.22 ± 2.22

Table 17 Full results for CART and OCT-H at depth 1.

Dataset				Mean out-of-sample accuracy		Mean improvement
Name	n	p	K	CART	OCT-H	
acute-inflammations-1	120	6	2	95.3	**100.0**	$+4.67 \pm 2.91$
acute-inflammations-2	120	6	2	**100.0**	**100.0**	0.00 ± 0.00
balance-scale	625	4	3	70.4	**87.6**	$+17.20 \pm 1.42$
banknote-authentication	1372	4	2	89.0	**98.7**	$+9.62 \pm 0.77$
blood-transfusion	748	4	2	**77.5**	77.4	-0.11 ± 0.68
breast-cancer-diagnostic	569	30	2	90.5	**94.0**	$+3.50 \pm 0.59$
breast-cancer-prognostic	194	32	2	**75.5**	**75.5**	0.00 ± 0.00
breast-cancer	683	9	2	93.1	**97.0**	$+3.86 \pm 1.01$
car-evaluation	1728	15	4	77.4	**87.5**	$+10.09 \pm 0.60$
chess-king-rook-vs-king-pawn	3196	37	2	90.6	**95.6**	$+4.98 \pm 0.63$
climate-model-crashes	540	18	2	91.0	**92.9**	$+1.93 \pm 0.69$
congressional-voting-records	232	16	2	**98.6**	**98.6**	0.00 ± 0.00
connectionist-bench-sonar	208	60	2	**70.8**	70.8	0.00 ± 2.02
connectionist-bench	990	10	11	34.4	**34.9**	$+0.49 \pm 0.32$
contraceptive-method-choice	1473	11	3	50.8	**51.1**	$+0.27 \pm 0.27$
credit-approval	653	37	2	87.7	**87.9**	$+0.12 \pm 0.12$
cylinder-bands	277	484	2	61.7	**65.5**	$+3.77 \pm 1.98$
dermatology	358	34	6	78.7	**83.4**	$+4.72 \pm 3.70$
echocardiogram	61	6	2	72.0	**77.3**	$+5.33 \pm 3.27$
fertility	100	12	2	86.4	**88.0**	$+1.60 \pm 0.98$
haberman-survival	306	3	2	**73.5**	73.0	-0.52 ± 0.52
hayes-roth	132	4	3	56.4	**71.5**	$+15.15 \pm 4.49$
heart-disease-cleveland	297	18	5	**54.9**	54.7	-0.27 ± 0.27
hepatitis	80	19	2	**82.0**	81.0	-1.00 ± 1.00
image-segmentation	210	19	7	52.5	**57.4**	$+4.91 \pm 2.50$
indian-liver-patient	579	10	2	71.6	**72.6**	$+0.97 \pm 0.68$
ionosphere	351	34	2	**87.8**	86.2	-1.61 ± 2.08
iris	150	4	3	92.4	**95.1**	$+2.70 \pm 1.71$
mammographic-mass	830	10	2	**81.7**	81.4	-0.39 ± 0.39
monks-problems-1	124	11	2	65.8	**93.5**	$+27.74 \pm 3.76$
monks-problems-2	169	11	2	60.9	**73.0**	$+12.09 \pm 7.11$
monks-problems-3	122	11	2	**94.2**	92.3	-1.94 ± 1.94
optical-recognition	3823	64	10	**41.4**	41.0	-0.38 ± 0.38
ozone-level-detection-eight	1847	72	2	**93.1**	**93.1**	0.00 ± 0.00
ozone-level-detection-one	1848	72	2	**96.8**	**96.8**	0.00 ± 0.00
parkinsons	195	21	2	**86.1**	84.9	-1.22 ± 0.82
pima-indians-diabetes	768	8	2	70.6	**71.4**	$+0.73 \pm 1.97$
planning-relax	182	12	2	**71.1**	69.8	-1.33 ± 0.89
qsar-biodegradation	1055	41	2	78.5	**83.0**	$+4.49 \pm 1.40$
seeds	210	7	3	86.8	**91.3**	$+4.53 \pm 1.75$
seismic-bumps	2584	20	2	**93.3**	**93.3**	0.00 ± 0.00
soybean-small	47	37	4	**100.0**	98.2	-1.82 ± 1.82
spambase	4601	57	2	86.0	**86.6**	$+0.59 \pm 1.24$
spect-heart	80	22	2	64.0	**70.0**	$+6.00 \pm 2.92$
spectf-heart	80	44	2	**69.0**	65.0	-4.00 ± 5.10
statlog-project-german-credit	1000	48	2	70.7	**71.0**	$+0.24 \pm 0.47$
statlog-project-landsat-satellite	4435	36	6	**77.7**	**77.7**	0.00 ± 0.00
teaching-assistant-evaluation	151	52	3	38.9	**55.1**	$+16.22 \pm 4.52$
thoracic-surgery	470	24	2	**85.5**	83.9	-1.54 ± 1.54
thyroid-disease-ann-thyroid	3772	21	3	**95.6**	92.5	-3.10 ± 0.06
thyroid-disease-new-thyroid	215	5	3	91.3	**95.8**	$+4.53 \pm 0.75$
tic-tac-toe-endgame	958	18	2	73.1	**97.0**	$+23.93 \pm 0.69$
wall-following-robot-2	5456	2	4	**100.0**	**100.0**	0.00 ± 0.00
wine	178	13	3	80.9	**92.9**	$+12.00 \pm 2.69$

Table 18 Full results for CART and OCT-H at depth 3.

Dataset				Mean out-of-sample accuracy		Mean improvement
Name	n	p	K	CART	OCT-H	
acute-inflammations-1	120	6	2	95.3	**100.0**	$+4.67 \pm 2.91$
acute-inflammations-2	120	6	2	**100.0**	**100.0**	0.00 ± 0.00
balance-scale	625	4	3	73.4	**87.6**	$+14.27 \pm 1.00$
banknote-authentication	1372	4	2	89.0	**98.7**	$+9.62 \pm 0.77$
blood-transfusion	748	4	2	**77.5**	77.4	-0.11 ± 0.68
breast-cancer-diagnostic	569	30	2	90.5	**94.0**	$+3.50 \pm 0.59$
breast-cancer-prognostic	194	32	2	**75.5**	**75.5**	0.00 ± 0.00
breast-cancer	683	9	2	93.1	**97.0**	$+3.86 \pm 1.01$
car-evaluation	1728	15	4	78.8	**87.5**	$+8.61 \pm 0.37$
chess-king-rook-vs-king-pawn	3196	37	2	90.4	**95.6**	$+5.21 \pm 0.75$
climate-model-crashes	540	18	2	91.0	**92.9**	$+1.93 \pm 0.69$
congressional-voting-records	232	16	2	**98.6**	**98.6**	0.00 ± 0.00
connectionist-bench-sonar	208	60	2	70.0	**71.5**	$+1.54 \pm 2.68$
connectionist-bench	990	10	11	**43.9**	**43.9**	0.00 ± 0.00
contraceptive-method-choice	1473	11	3	**53.8**	53.3	-0.49 ± 0.49
credit-approval	653	37	2	87.5	**87.9**	$+0.37 \pm 0.25$
cylinder-bands	277	484	2	62.6	**65.5**	$+2.90 \pm 1.45$
dermatology	358	34	6	89.2	**92.6**	$+3.37 \pm 0.94$
echocardiogram	61	6	2	72.0	**77.3**	$+5.33 \pm 3.27$
fertility	100	12	2	**88.0**	**88.0**	0.00 ± 0.00
haberman-survival	306	3	2	**72.7**	**72.7**	0.00 ± 0.00
hayes-roth	132	4	3	56.4	**70.3**	$+13.94 \pm 4.35$
heart-disease-cleveland	297	18	5	54.9	**55.5**	$+0.53 \pm 0.53$
hepatitis	80	19	2	**82.0**	81.0	-1.00 ± 1.00
image-segmentation	210	19	7	64.2	**66.0**	$+1.89 \pm 1.89$
indian-liver-patient	579	10	2	69.8	**72.6**	$+2.76 \pm 1.68$
ionosphere	351	34	2	**87.8**	86.2	-1.61 ± 2.08
iris	150	4	3	92.4	**95.1**	$+2.70 \pm 1.71$
mammographic-mass	830	10	2	**81.7**	81.4	-0.39 ± 0.39
monks-problems-1	124	11	2	68.4	**93.5**	$+25.16 \pm 4.93$
monks-problems-2	169	11	2	62.8	**73.0**	$+10.23 \pm 7.52$
monks-problems-3	122	11	2	**94.2**	92.3	-1.94 ± 1.94
optical-recognition	3823	64	10	**54.7**	54.3	-0.38 ± 0.38
ozone-level-detection-eight	1847	72	2	**93.1**	**93.1**	0.00 ± 0.00
ozone-level-detection-one	1848	72	2	**96.8**	**96.8**	0.00 ± 0.00
parkinsons	195	21	2	**86.1**	84.9	-1.22 ± 0.82
pima-indians-diabetes	768	8	2	**71.7**	70.3	-1.35 ± 2.70
planning-relax	182	12	2	**71.1**	68.4	-2.67 ± 1.78
qsar-biodegradation	1055	41	2	79.6	**84.0**	$+4.41 \pm 1.46$
seeds	210	7	3	86.8	**91.3**	$+4.53 \pm 1.75$
seismic-bumps	2584	20	2	**93.3**	**93.3**	0.00 ± 0.00
soybean-small	47	37	4	**100.0**	98.2	-1.82 ± 1.82
spambase	4601	57	2	86.0	**86.6**	$+0.59 \pm 1.24$
spect-heart	80	22	2	64.0	**70.0**	$+6.00 \pm 2.92$
spectf-heart	80	44	2	**70.0**	65.0	-5.00 ± 5.92
statlog-project-german-credit	1000	48	2	69.9	**71.0**	$+1.04 \pm 1.14$
statlog-project-landsat-satellite	4435	36	6	**78.2**	**78.2**	0.00 ± 0.00
teaching-assistant-evaluation	151	52	3	45.4	**56.2**	$+10.81 \pm 4.01$
thoracic-surgery	470	24	2	**85.5**	83.9	-1.54 ± 1.54
thyroid-disease-ann-thyroid	3772	21	3	**95.6**	92.5	-3.10 ± 0.06
thyroid-disease-new-thyroid	215	5	3	91.3	**95.8**	$+4.53 \pm 0.75$
tic-tac-toe-endgame	958	18	2	74.2	**97.0**	$+22.76 \pm 1.03$
wall-following-robot-2	5456	2	4	**100.0**	**100.0**	0.00 ± 0.00
wine	178	13	3	80.9	**91.6**	$+10.67 \pm 2.85$

Table 19 Full results for CART and OCT-H at depth 4.