Optimally Trained Classification Trees and Occam's Razor

Savický, Petr
2001

Institute of Computer Science
Academy of Sciences of the Czech Republic

Optimally trained classification trees and Occam's razor

P. Savický, J. Klaschka

Technical report No. 855

December 2001

Pod Vodárenskou věží 2, 182 07 Prague 8, phone: (+4202) 6605 1111, fax: (+4202) 858 57 89,
e-mail:savicky@cs.cas.cz, klaschka@cs.cas.cz

Institute of Computer Science
Academy of Sciences of the Czech Republic

Optimally trained classification trees and Occam's razor

P. Savický, J. Klaschka

Technical report No. 855

December 2001

Abstract:

Two algorithms constructing the trees with a minimum misclassification error on the training data are described. Results of numerical experiments focused on the comparison of these algorithms with the classical tree construction by recursive partitioning are presented. For this purpose, several explicitly given distributions were used. The trees are compared from the point of view of their generalization properties, i.e. their error on unseen cases from the same distribution, not used for training. For some of the distributions, the trees from the optimization algorithms clearly outperform those from recursive partitioning. For other distributions, however, the situation is the opposite. The problems where optimization yields worse results than CART are the main focus of attention. Since the size and error on training data of the trees obtained by optimization do not exceed the corresponding parameters of the trees obtained by classical methods, the result provides new examples of situations, where the preference of small models suggested by Occam's razor principle does not improve the quality of the model.

Keywords:
Classification trees, recursive partitioning, optimization, dynamic programming, bottom-up algorithms, generalization, Occam's razor

Acknowledgement. The research of both authors was supported by the Grant Agency of the Czech Republic, grant no. 201/00/1482.

1 Introduction

Classification trees have traditionally been grown in a top-down manner, by a "local" optimization of individual splits, so called recursive partitioning. Savický et al. [9] proposed two bottom-up algorithms aimed at the full optimization of classification trees. The algorithms work for a limited number of binary predictors, and their implementation is aimed, apart from other things, at studying the difference between full optimization and recursive partitioning.

The algorithms construct trees of various sizes, and each of the resulting trees attains the minimum classification error (or cost) on the training data given the size. We call such trees *optimally trained trees*. The common tree growing methodologies are based on a kind of search for trees with good trade-off between size and error. Consequently, the optimally trained trees may be expected to have good generalization properties, i.e. a small error (cost) also on independent data drawn from the same distribution, not used for training. In some cases, and in particular for the experiments reported in Savicky et al. [9], this belief is satisfied.

There are, however, also cases, and the present paper is focused on them, where optimally trained trees are worse than the trees grown by recursive partitioning. By more detailed analysis of these situations, we obtain new examples of situations, where the preference of small models suggested by Occam's razor principle does not improve the quality of the model. The experimental results demonstrating this phenomenon contribute to the current discussion on the extent to which the Occam's razor principle is applicable in machine learning, see e.g. [3], [8], [10]. Our experiments are conceptually similar to those of [8], but for a more complex situation. Using a substantially more efficient algorithm for tree optimization, we work with much larger trees, which, moreover, need not be exact trees, i.e. may have nonzero error on the training set.

After preliminaries (basics of the tree-based methodologies) in Section 2 and remarks on the Occam's razor principle in Section 3, we describe the algorithms growing optimally trained trees in Section 4. In Section 5 we deal with our computational experiments. The distributions used and the structure of the experiments are described and the results of the experiments are briefly outlined. The detailed numerical results are left to Appendix A. Some remarks concluding the paper may be found in Section 6.

2 Decision trees for classification problems

Assume a probability distribution D on $(n+1)$-tuples $(x_1, ..., x_n, y)$, where x_i are predictors (numerical or categorical) and y is the class (element of a finite set C). A classifier is a function f, such that $f(x_1, ..., x_n)$ is a reasonable prediction of y, given $x_1, ..., x_n$. At the same time, by a classifier we also mean a representation of such a function in the form of, e.g., tree diagram, list of discrimination equations, etc. (Note that a function may have multiple representations, while it is uniquely determined by any of its representations.) Whenever we use, throughout this paper, symbol f with possible subscripts, we mean a classifier as a function, while T, again with possible subscripts, will denote a tree representation of such a function. The function represented by T will be denoted f_T. A classifier is assumed to be used in situations, where the predictors x_i may be directly observed, while the class y may not.

In order to measure the error of a classifier T, we use misclassification costs $c(i, j)$ as the penalty for classifying a class i case as j. Let $p(i, j)$ be the probability that a case from D has class i and is classified as j, i.e. $p(i, j) = P(y = i \wedge f_T(x_1, ..., x_n) = j)$. The error of a classifier T is measured by the expected cost of its predictions, which is

$$R_D(T) = \sum_{i,j \in C} c(i, j) \cdot p(i, j).$$

This formula is also used to determine the sample estimates of the expected cost based on a data set $\mathcal{L}$. For this, let $p_{\mathcal{L}}(i,j)$ be sample estimates of the true probabilities based on data set $\mathcal{L}$. Then

$$R_{\mathcal{L}}(T) = \sum_{i,j \in C} c(i,j) \cdot p_{\mathcal{L}}(i,j).$$

The simplest form of the estimates $p_{\mathcal{L}}(i,j)$ is

$$p_{\mathcal{L}}(i,j) = \frac{m(i,j)}{m}, \tag{1}$$

where $m(i,j)$ is the number of cases from class i classified as j, and m is the overall number of cases in $\mathcal{L}$. Note that this reduces to the relative number of misclassified cases in $\mathcal{L}$ when $c(i,j) = 1 - \delta_{ij}$, where δ_{ij} is Kronecker delta, which is a common setting. The estimates (1) are proper when the data in $\mathcal{L}$ are the only source of information on $p_D(i,j)$. Often it is not the case. The joint probability $p_D(i,j)$ of the true and predicted classes may be expressed as the product of the marginal probability $p_D(i)$ of true class i (so called prior probability, or, briefly, prior), and the conditional probability $p_D(j|i)$ of the predicted class j given the true class i. When an "external" information on the prior probabilities is available, so that $p_D(i)$ is estimated with a number π_i, the proper estimate of $p_D(i,j)$ is

$$p_{\mathcal{L}}(i,j) = \pi_i \cdot \frac{m(i,j)}{m(i)}, \tag{2}$$

where $m(i)$ is the number of class i cases in $\mathcal{L}$. Note that (2) reduces to (1) when $\pi_i = m(i)/m$.

By a tree classifier with univariate splits, we understand a binary tree, whose internal nodes are labeled by tests based on single predictors and whose leaves are labeled by classes. For a numerical predictor x_i, the standard univariate test has the form $x_i \leq a$ for a real number a, while for a categorical predictor, it has the form $x_i \in A$, where A is a subset of the finite range of x_i. The two successors of a test node correspond to the positive and negative answers to the test question. A tree T represents a function $f_T(x_1, ..., x_n)$, which is defined for every vector $x_1, ..., x_n$ of predictors as follows. Start in the root and in each test node, continue to the successor determined by the answer to the test question. The value of the leaf finally reached is $f_T(x_1, ..., x_n)$. Let $\tilde{T}$ be the set of leaves of T.

We investigate the methods of construction of tree based classifiers on the basis of a sample $\mathcal{L}$ of m independent cases from D, which may be considered as an $m \times (n+1)$ matrix, whose rows correspond to the observed cases. The first n columns of the matrix correspond to the predictors and the last column corresponds to the dependent variable.

The standard methodologies of tree construction, such as CART [2], C4.5 [6], QUEST [7] and many others, are based on recursive partitioning. The general structure of recursive partitioning applied to a learning sample $\mathcal{L}$ consists in iterated splitting of the data. In each step, the data are split by selecting an appropriate test and the two parts are formed by cases satisfying resp. not satisfying the test condition. Then, the procedure is applied recursively to the two subsamples corresponding to the two successors, etc. The criteria for selecting the test condition are such that splits with larger difference in distribution of classes between the two successors are preferred. The tree based methods may be divided into two groups, according to the way of how the process of splitting is stopped and a "final" tree selected. The simpler approach is based on a relatively strict stopping criterion which requires that each split must be effective enough. When no such split is found for a node, the splitting of the node is stopped. The tree is completed when the splitting of all the leaves is stopped. Another, more sophisticated approach consists in growing first a big tree, using a week stopping criterion, and on a subsequent pruning. The extent of pruning is given by estimates of true errors of trees of different sizes. For more details on the methods and their applications, see e.g. [2], [6], [4].

The goal of the above strategy is to produce a tree, such that in most of its leaves one of the classes has clear majority. Although there is no general guarantee that the goal is reached, it indeed happens in a lot of practically important cases.

3 Applicability of Occams's razor principle

In decision tree induction, one has to look for trees which are both reasonably small and accurate on training data. It is important that this requirement includes to minimize also the size. A large tree can reach a small error on training data by adapting not only to the general dependence in the data, but also, too closely, to the specific selection of examples that happen to be present in the training set and to cases affected by noise. Such a tree may have much larger error on unseen cases than on training data. This phenomenon is called overfit, and the typical way to avoid it is to prefer smaller trees, even if they have a bit larger error on training data. The typical way of doing that is pruning, see e.g. [2], [6], [4].

Let us point out that the preference of small models is also supported by PAC learning theory, see e.g. [1]. The result is that if a model described by a bit string of length s correctly classifies k out of m independent cases from a distribution D and $\alpha = 2^s \binom{m}{k} (1-\varepsilon)^k \varepsilon^{m-k} < 1$, then one can make the conclusion that the probability of the correct classification on D is at least $1-\varepsilon$ on the level α of statistical significance. If $k > (1-\varepsilon)m$ and s is a small fraction of m (depending on the difference between k and $(1-\varepsilon)m$), then α may be exponentially small and, hence, the conclusion may be quite reliable. This way, however, we obtain for smaller models a higher *lower bound* of the probability of the correct classification, not a higher probability in itself. Thus, the PAC learning theory may encourage the preference of small models, but not to prove that such a strategy is always helpful.

Although the exact conditions under which the preference of small models helps are not known, this preference is a widely used and helpful methodological suggestion. It is usually called Occam's razor principle, since its nature is closely related to a well-known criterion for theories in natural sciences formulated by W. Occam in 14th century as a criticism of scholastic philosphy. For more details see e.g. [3] and the references therein.

Besides the classical interpretation of Occam's razor principle, it is also possible to consider another principle, which is dual to it. There is typically a trade-off between complexity of the sought model and its accuracy. Consider the region of the reachable points whose x-coordinate in a two dimensional diagram is the size of the model and y-coordinate is its error on training data. This region is usually bounded below by a nonincreasing curve. If it is, moreover, strictly decreasing, then the requirements to minimize either the complexity or the error while preserving the other parameter, are both equivalent to restricting to the points on the curve. Hence, we can also use the following dual principle: "Among models of given complexity choose a model of the smallest error on the training data". This is a frequently used approach.

Despite the fact that the preference of small (more accurate) trees *typically* helps to achieve a good generalization, there is no guarantee that it *always* helps. Moreover, since the size of the tree and its error on training data are not the only parameters determining its accuracy on the true distribution, one should expect that there are situations, where these criteria indeed lead to worse results than some other criteria.

There are experimental results demonstrating examples of situations, where smaller models can be worse.

Murphy and Pazzani [8] consider an artificial two class problem with 5 binary predictors ($x_1x_2x_3 \vee x_4x_5$). In 100 independent trials, they consider a training set $\mathcal{L}_1$ of 20 cases and a testing set $\mathcal{L}_2$ of 12 cases (the complement). For each pair $\mathcal{L}_1$ and $\mathcal{L}_2$ they construct all those trees of sizes up to 10 that exactly classify $\mathcal{L}_1$. For trials, where the smallest exact trees have sizes 4 and 6, the classifiers of sizes 5 and 7, resp., have smaller errors on average.

Webb [10] suggested an extension of the tree-growing algorithm C4.5, which is called C4.5X. This extended algorithm adds to C4.5 a postprocessing phase, which extends the tree resulting from C4.5 with further nonredundant nodes. The new tree does the same classification of the training data, but frequently has an improved classification of unseen cases.

4 Tree optimization

In this section, we describe two tree optimization algorithms suggested in [9]. They are designed, besides other things, for experiments aimed at studying the difference between recursive partitioning and tree optimization.

Definition 4.0.1 The cost-complexity measure of T with a real valued parameter $\alpha \geq 0$ on a training data set $\mathcal{L}$ is $C_{\mathcal{L},\alpha}(T) = R_{\mathcal{L}}(T) + \alpha|\tilde{T}|$.

Definition 4.0.2
1. Tree is α-cost-complexity optimal on $\mathcal{L}$, if it has the minimum value of $C_{\mathcal{L},\alpha}(T)$ among all trees.
2. Tree T is optimally trained on $\mathcal{L}$, if it has minimum $R_{\mathcal{L}}(T)$ among all trees of the same $|\tilde{T}|$.

Clearly, any α-cost-complexity optimal tree for $\alpha \geq 0$ is also optimally trained. The reverse is not true.

We describe two algorithms called Algorithm 1 and 2. Algorithm 1 finds a tree which is α-cost-complexity optimal for a given $\alpha \geq 0$. Algorithm 2 finds an optimally trained tree for every size from one up to a given limit. Both algorithms use the dynamic programming approach. The original idea was suggested by [5] in connection with the smallest exact representation of a function on $\{0,1\}^n$ by a tree. It was adapted for constructing trees with nonzero classification error and for data sets, whose predictor vectors may not cover the whole set $\{0,1\}^n$.

Algorithms 1 and 2 may be applied to problems with a limited number of binary predictors. In this case, the predictor space of the considered distribution is the Boolean cube $\{0,1\}^n$. The idea of the algorithms is to construct optimal trees for all subcubes of $\{0,1\}^n$ containing at least one training case, where a subcube of $\{0,1\}^n$ is a subset of $\{0,1\}^n$ characterized by restricting some variables to constants. This requires to extend slightly the concept of tree: We will understand by a tree not only a "total classifier" assigning a class to *any* predictor vector, but also a "partial classifier" that classifies only those cases that belong to a specific subcube. In the latter case, the domain is a part of the description of the tree.

Every subcube may be represented by a string in $\{0,1,*\}^n$. Symbols 0,1 represent the values to which the corresponding variables are restricted. Symbol $*$ represents free (unrestricted) variables. It follows that there are 3^n subcubes of $\{0,1\}^n$. The number of free variables in a subcube is called the dimension of the subcube. Clearly, a subcube of dimension k contains 2^k elements.

For a subcube s and a tree T testing only variables that are free in s, let (s,T) represent tree T considered as a representation of a function with domain s. We say that (s,T) is formed by subtrees (s_1,T_1) and (s_2,T_2), if the subcube s is a disjoint union of subcubes s_1, s_2, the root of T tests the (unique) variable which is set to different values in s_1 and s_2, and the two subtrees of T are equal to T_1 and T_2 respectively.

For any pair (s,T) and a distribution D on $\{0,1\}^n \times C$, we can define $R_D(s,T)$ by

$$R_D(s,T) = \sum_{i,j \in C} c(i,j) \cdot p_D(s,i,j), \tag{3}$$

where $p_D(s,i,j)$ is the probability that a case drawn at random from D has its predictor vector in s, belongs to class i, and is classified by T to class j. Similarly, for a data set $\mathcal{L}$, $R_{\mathcal{L}}(s,T)$ is defined as

$$R_{\mathcal{L}}(s,T) = \sum_{i,j \in C} c(i,j) \cdot p_{\mathcal{L}}(s,i,j), \tag{4}$$

where $p_{\mathcal{L}}(s,i,j)$ is an estimate of $p_D(s,i,j)$, given by either

$$p_{\mathcal{L}}(s,i,j) = \frac{m(s,i,j)}{m},$$

or

$$p_{\mathcal{L}}(s,i,j) = \pi_i \, m(s,j|i) = \pi_i \cdot \frac{m(s,i,j)}{m(i)}$$

where $m(s,i,j)$ is the number of those class i cases classified as j, whose predictor vector $(x_1,\ldots,x_n)$ belongs to s, while $m(i)$ and m have the same meaning as in (1) and (2). The choice of the estimate of $p_D(s,i,j)$ depends of whether there is an "external" information on priors, or not. Obviously,

$$R_D(s,T) = R_D(s_1,T_1) + R_D(s_2,T_2)$$

and

$$R_{\mathcal{L}}(s,T) = R_{\mathcal{L}}(s_1,T_1) + R_{\mathcal{L}}(s_2,T_2)$$

for a tree (s,T), which is formed by subtrees (s_1,T_1), (s_2,T_2).

Now, the definitions of optimality of the trees may be formulated for the individual subcubes.

Definition 4.0.3 The cost-complexity measure of (s,T) with parameter $\alpha \geq 0$ on a training data set $\mathcal{L}$ is $C_{\mathcal{L},\alpha}(s,T) = R_{\mathcal{L}}(s,T) + \alpha|\tilde{T}|$.

Definition 4.0.4

1. Tree (s,T) is α-cost-complexity optimal on $\mathcal{L}$, if $C_{\mathcal{L},\alpha}(s,T) \leq C_{\mathcal{L},\alpha}(s,T')$ for all trees (s,T').
2. Tree (s,T) is optimally trained on $\mathcal{L}$, if $R_{\mathcal{L}}(s,T) \leq R_{\mathcal{L}}(s,T')$ for all trees (s,T') such that $|\tilde{T}'| = |\tilde{T}|$.

Again, any α-cost-complexity optimal tree for $\alpha \geq 0$ is also optimally trained.

For simplicity of formulating the algorithms, let $m(\{\bar{x}\},i)$ be the number of cases in $\mathcal{L}$ with predictor vector $\bar{x} \in \{0,1\}^n$ and class $i \in C$.

4.1 Algorithm 1

Input: training data set $\mathcal{L}$, $\alpha \geq 0$, priors π_i, costs $c(i,j)$.
Output: α-cost-complexity optimal tree T.

Calculate the table of $m(\{\bar{x}\},i)$ for all $\bar{x} \in \{0,1\}^n$ and $i \in C$. This table is used in all the remaining steps of the algorithm instead of the direct use of $\mathcal{L}$.

Consider all subcubes of $\{0,1\}^n$ in an order such that all cubes strictly contained in a subcube s are considered before s.

For each subcube s construct the tree (s,T_s) using results for smaller subcubes as follows:

1. Determine the minimum $C_{\mathcal{L},\alpha}(s,T)$ among the trees (s,T) with only one leaf, i.e. choose for s the best constant from C.

2. For each j such that x_j is a free variable in s, determine $C_{\mathcal{L},\alpha}(s,T)$ of the tree (s,T), which tests x_j in its root and is formed by trees (s_L, T_{s_L}), (s_R, T_{s_R}) constructed for subcubes s_L and s_R.
3. (s, T_s) is the tree with the minimum $C_{\mathcal{L},\alpha}(s,T)$ among the trees (s,T) considered in the two previous steps.

The result of Algorithm 1 is the tree (s, T_s) for s equal to the whole set $\{0,1\}^n$.

Lemma 4.0.1 *For every subcube s, tree (s, T_s) constructed by the above procedure is an α-cost-complexity optimal tree.*

Proof. Let us proceed by induction on the dimension of the subcube s. The statement is clearly true when there are no free variables in s. Otherwise, let (s, T') be any tree on the domain s.

If $|\tilde{T}'| = 1$, then $C_{\mathcal{L},\alpha}(s,T') \geq C_{\mathcal{L},\alpha}(s,T_s)$, since Algorithm 1 takes all constant trees on s into account, while selecting T_s.

Let $|\tilde{T}'| > 1$, let x_j be the variable tested in the root of T' and let (s_L, T'_L), (s_R, T'_R) be the successors of the root in T'. Consider the trees (s_L, T_{s_L}) and (s_R, T_{s_R}) constructed by Algorithm 1 for s_L and s_R. By the induction hypothesis, these trees are α-cost-complexity optimal. Hence, we have $C_{\mathcal{L},\alpha}(s,T') = C_{\mathcal{L},\alpha}(s_L,T'_L) + C_{\mathcal{L},\alpha}(s_R,T'_R) \geq C_{\mathcal{L},\alpha}(s_L,T_{s_L}) + C_{\mathcal{L},\alpha}(s_R,T_{s_R})$. Since the tree formed by (s_L,T_{s_L}) and (s_R,T_{s_R}) belongs to the set of candidates considered while selecting T_s, we have $C_{\mathcal{L},\alpha}(s,T') \geq C_{\mathcal{L},\alpha}(s,T_s)$. □

4.2 Algorithm 2

Input: training data set $\mathcal{L}$, integer M, priors π_i, costs $c(i,j)$.
Output: optimally trained classification trees $T_1, \ldots, T_M$, where T_k has size k.

Calculate the table of $m(\{\bar{x}\}, i)$ for all $\bar{x} \in \{0,1\}^n$ and $i \in C$. This table is used in all the remaining steps of the algorithm instead of the direct use of $\mathcal{L}$.

Consider all subcubes s in an order such that all cubes contained in s are considered before s. Construct $(s, T_{s,k})$ for all $k = 1, \ldots, M$ for which the construction is possible using the results for smaller subcubes as follows:

If $k = 1$, $(s, T_{s,k})$ is the tree (s,T) with the smallest $R_{\mathcal{L}}(s,T)$ among all one node trees.
If $k \geq 2$, do the following:

1. For all partitions $k = k_L + k_R$ of the size of the tree and every variable x_j that is free in s, determine the error of (s,T), where T tests x_j in its root and is formed by (s_L, T_{s_L,k_L}) and (s_R, T_{s_R,k_R}) constructed by the algorithm for subcubes s_L and s_R.
2. Find and store as $(s, T_{s,k})$ the tree (or one of such trees, if more exist) with the minimum error on $\mathcal{L}$ among the trees (s,T) considered in the previous step, if there is any.

The result of Algorithm 2 is the sequence of those trees $T_{s,k}$ for $k = 1, \ldots, M$, that were successfully constructed, where s is equal to the whole set $\{0,1\}^n$.

Lemma 4.0.2 *For every s and k, tree $(s, T_{s,k})$ constructed by the above procedure is optimally trained.*

Proof. The statement is clearly true for $k = 1$, since all constant trees are considered while selecting $T_{s,k}$.

Let us prove the statement for $k \geq 2$ by induction on the dimension of the subcube s. If there are no free variables in s and $k \geq 2$, no tree is constructed. Let the dimension

of s be nonzero, $k \geq 2$, and let (s, T') be any tree on the domain s with $|\tilde{T}'| = k$. Let x_j be the variable tested in the root of T', let (s_L, T'_L), (s_R, T'_R) be the successors of the root in T', and k_L, k_R their sizes, resp. Consider the trees (s_L, T_{s_L,k_L}) and (s_R, T_{s_R,k_R}) constructed by Algorithm 2 for subcubes s_L, s_R and sizes k_L, k_R, resp. By the induction hypothesis, these trees are optimally trained. Hence, we have $R_{\mathcal{L}}(s, T') = R_{\mathcal{L}}(s_L, T'_L) + R_{\mathcal{L}}(s_R, T'_R) \geq R_{\mathcal{L}}(s_L, T_{s_L,k_L}) + R_{\mathcal{L}}(s_R, T_{s_R,k_R})$. Since the tree formed by (s_L, T_{s_L,k_L}) and (s_R, T_{s_R,k_R}) belongs to the set of candidates considered while selecting $T_{s,k}$, we have $R_{\mathcal{L}}(s, T') \geq R_{\mathcal{L}}(s, T_{s,k})$. □

4.3 Implementation and complexity of the algorithms

Both Algorithms 1 and 2 are implemented in a single C++ program. For each of 3^n subcubes, Algorithms 1 uses $O(|C|)$ of memory cells and tests $O(n)$ variants, each of them in constant time. Thus, the space and time complexities are $O(|C|3^n)$ and $O(n3^n)$, resp. For Algorithm 2, the memory reserved for a subcube is $O(|C|M)$, and the number of variants to be tested is $O(nM)$. This leads to space and time complexities $O(|C|3^n M)$ and $O(n3^n M)$,resp. The factor 3^n in all the expressions may be improved to $|\mathcal{L}|2^n$ by storing information only for subcubes with at least one training case.

The optimization algorithms can hardly be utilized for problems of practical interest. Algorithm 2, however, may be used e.g. for 10 binary predictors and $M = 160$ using easily available hardware. This is sufficient for nontrivial examples.

For the experiments described in the next Section, we used 8 nodes (PC, 400 MHz CPU, 380MB RAM) of a Linux cluster. Analysis of 800 training-testing data set pairs by Algorithm 2 required approximately 9 hours.

5 Experiments

We performed extensive computational experiments with artificial data in order to compare generalization properties of optimally trained trees with those of the trees constructed by recursive partitioning. The optimally trained trees were constructed by Algorithm 2 which, according to our previous experience (see e.g. [9]) yields in most cases slightly better results than Algorithm 1. (The advantage of Algorithm 1 is in higher execution speed and smaller memory required.) CART [2] was used for comparison as a representative of classical tree-based methods.

Each of the distributions used in the experiments can be verbally characterized as a Boolean function contaminated by an attribute noise, and by irrelevant noise variables. One of the distributions, namely the distribution based on the parity function, is specifically designed for the demonstration of the virtue of the optimization approach. The rest of the distributions, being based on monotonous functions, represents "the opposite end of the scale".

There were three different modes of experiment, referred to as Experiment I, II, and III. They differ from each other in the way of how a single "best" tree is selected from among a series of optimally trained trees of different sizes. The modes cited are defined in paragraphs 5.2, 5.3 and 5.4. Experiment I, where the size selection procedure is effectively the same as in the "real-life" data analyses, was entered by all the distributions studied. Experiments II and III were more specifically related to the search for an evidence against the Occam's razor principle. Therefore the distribution based on the parity function was omitted from Experiments II and III.

5.1 Distributions used for the experiments

The experiments reported in this paper use distributions defined using Boolean functions. The exact description of the distributions is as follows: Let n be the total number of predictors and $m \leq n$ be the number of relevant predictors. Let $z_1, ..., z_n$ be auxiliary (hidden) random variables distributed uniformly and independently on $\{0,1\}$. Let $\theta_1, \ldots, \theta_n$ be independent random variables on $\{0,1\}$ satisfying $P(\theta_i = 1) = \nu$, where ν controls the amount of noise. A case from D is generated as $(x_1, \ldots, x_n, y) = (z_1 \oplus \theta_1, z_n \oplus \theta_n, g(z_1, ..., z_m))$, where $\oplus$ is addition modulo two.

All the reported experiments used $n = 10$ predictors and some of the functions $g_1, \ldots, g_4$ described below as the function g. The most detailed description of the results is given for experiments with g_1 (parity function) and g_2 (majority function). For g_3 (called two thresholds) and g_4 (called two squares) only a few numerical results are presented.

Let $g_1, \ldots, g_4$ be defined by

$$\begin{aligned} g_1(x_1, \ldots, x_5) &= x_1 \oplus x_2 \oplus \ldots \oplus x_5, \\ g_2(x_1, \ldots, x_7) &= T_4^7(x_1, \ldots, x_7), \\ g_3(x_1, \ldots, x_8) &= T_3^4(x_1, \ldots, x_4) \vee T_3^4(x_5, \ldots, x_8), \\ g_4(x_1, \ldots, x_8) &= Q_{10}\big((x_1 + \cdots + x_4)^2 + (x_5 + \cdots + x_8)^2\big), \end{aligned}$$

where T_k^m is a Boolean function satisfying

$$T_k^m(x_1, \ldots, x_m) = 1 \Leftrightarrow \sum_{i=1}^{m} x_i \geq k,$$

and Q_k is a 0/1-valued function defined by

$$Q_k(x) = 1 \Leftrightarrow x \geq k.$$

Exact prior probabilities of the distributions were used by both Algorithm 2 and CART. The misclassification costs $c(i,j)$ were set to $1-\delta_{ij}$, where δ_{ij} is the Kronecker delta. Thus, the true misclassification cost of a tree reduces to the probability of incorrect classification.

Samples from distributions D for all parameter settings were generated using pseudo-random generator MRG32k5a by P. L'Ecuyer [11], see also [12].

5.2 Experiment I

The first experiment is aimed at comparison of tree optimization and recursive partitioning represented by CART [2] with Gini criterion and 0SE selection rule. The parameters of CART were selected on the basis of our preliminary experiments, where these parameters performed well. The experiment is parametrized by a distribution D, the size ℓ of the sample used for tree construction, the maximum size M of constructed trees, and consists of the following steps:

1. Construct two independent samples $\mathcal{L}_g$ and $\mathcal{L}_v$ from D of sizes $|\mathcal{L}_g| = \ell$ and $|\mathcal{L}_v| = \text{round}(\ell/2)$.
2. Construct the optimally trained trees of all sizes up to M on L_g. Among these trees find the tree with the smallest error on L_v. In the case of a tie, choose the smaller one. Denote this tree T_{opt}.
3. Use CART to grow a tree using $\mathcal{L}_g$ for growing and $\mathcal{L}_v$ for validation (or "test sample for pruning", using terminology of CART). Denote the resulting tree T_{cart}.
4. Determine the true classification errors of T_{opt} and T_{cart} using the information on the distribution D.

The errors of T_{opt} and T_{cart} are random variables depending on L_g and L_v. Repeating the procedure 100 times, we obtained 100 independent realizations of each of these

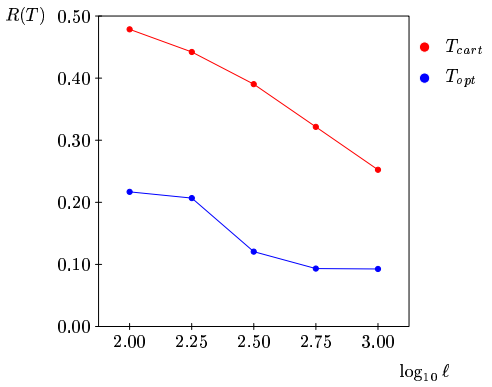

Figure 1: Experiment I with parity ($g = g_1$), attribute noise level $\nu = 0.02$. Mean true errors of 100 trees grown by Algorithm 2 and by CART for different sizes ℓ of training data sets.

random variates. The expected errors of T_{opt} and T_{cart} were estimated by arithmetical means. Moreover, standard errors of the estimates were calculated, and a possible difference between the expected errors was tested using the two sided paired Student's t-test. We performed the above experiment several times for different values of ℓ, using independent samples for each ℓ. As a result, for each size ℓ of L_g, we could compare the expected errors of T_{opt} and T_{cart} in terms of both descriptive statistics, and statistical significance.

5.2.1 Results for parity

We use the function $g = g_1$, the amount $\nu = 0.02$ of attribute noise, and sizes $\ell = \text{round}(10^2), \text{round}(10^{2.25}), \ldots, \text{round}(10^3)$, i.e. $\ell = 100, 178, ..., 1000$. The maximum size of constructed trees is $M = 100$. The smallest exact tree for g_1 has size 32 and the sizes of tree T_{opt} do not exceed 50 for data sizes $\ell \geq 362$ (while for $\ell = 100$ and $\ell = 178$ there is a large variability of tree sizes, and some trees of sizes close to or equal to 100 occur).

The result of the experiment is that optimally trained trees clearly outperform those from CART. This is demonstrated by the estimated average errors shown at Fig. 1. An explanation is as follows. Splitting on a single relevant predictor (an argument of the parity function) does not help to separate classes 0 and 1. The parity function may be discovered only with a deeper "look-ahead", which is a key feature of the optimization algorithm. The recursive partitioning is easily misled to split on the noise variables, which changes the task of finding the underlying structure in one data set to several analogous tasks with smaller data sets. This is, of course, harder and, hence, leads to very large and imprecise trees.

Detailed tables of numerical results can be found in Appendix A.1.

5.2.2 Results for majority

We use the function $g = g_2$, the amount $\nu = 0.1$ of attribute noise and sizes $\ell = \text{round}(10^2), \text{round}(10^{2.25}), \ldots, \text{round}(10^{3.75})$, i.e. $\ell = 100, 178, ..., 5623$. The maximum size of constructed trees is $M = 160$. The smallest exact tree for g_2 has size 70, and sizes of trees T_{opt} over 120 are infrequent (5% at most) for $\ell \geq 562$ (9% for $\ell = 316$, 21% for $\ell = 178$).

The optimally trained trees appeared to give worse generalization than CART for $\ell \geq 316$, as demonstrated by Fig. 2, and the tables in Appendix A.2.

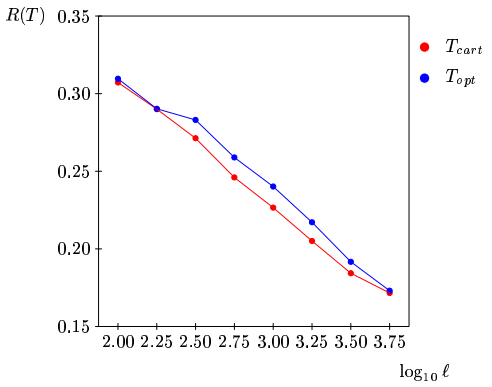

Figure 2: Experiment I with majority ($g = g_2$), attribute noise level $\nu = 0.1$. Mean true errors of 100 trees grown by Algorithm 2 and by CART for different sizes ℓ of training data sets.

5.2.3 Results for two thresholds

Function $g = g_3$, the noise levels $\nu = 0.02$ and 0.1, and sizes $\ell = \mathrm{round}(10^2), \mathrm{round}(10^{2.25}), \ldots, \mathrm{round}(10^{3.75})$ were used. Trees of sizes up to 160 were constructed by Algorithm 2. The smallest exact tree for g_3 has size 61.

For the smaller noise level, i.e. $\nu = 0.02$, the trees resulting from Algorithm 2 performed worse than those from CART for sizes from 562 to 3162, while for the higher amount of noise, $\nu = 0.1$, a similar result was observed in an even broader range of sizes, from 316 to 5623.

The results are summarized in Appendices A.3 and A.4.

5.2.4 Results for two squares

Function $g = g_4$, the noise levels $\nu = 0, 0.02$ and 0.1, and sizes $\ell = \mathrm{round}(10^2), \mathrm{round}(10^{2.25}), \ldots, \mathrm{round}(10^{3.75})$ were used. Trees of sizes up to 160 were constructed by Algorithm 2. The smallest exact tree for g_3 has size 88.

For each noise level, the trees grown by Algorithm 2 had, on average, worse generalization than those from CART for some data sizes, and the range of such sizes was the wider, the higher was the amount of noise. Thus, for $\nu = 0$ (i.e. without the attribute noise), the phenomenon was observed at sizes 562 and 1000, for $\nu = 0.02$ the range was extended to 316-3162, and for $\nu = 0.1$ to 178-5623.

The results are summarized in Appendices A.5 to A.7.

5.3 Experiment II

In experiment I, there is no control on the relationship between the sizes of T_{opt} and T_{cart}. They are typically different and any of them may be larger. In order to make stronger conclusion on the basis of our experiments, we introduced another setup, in which the tree optimization is forced to use a tree of the same size as CART. In this case, $\mathcal{L}_v$ is not used in the tree optimization part and, hence, we do not use it also in CART to achieve a fair comparison. The required validation sets for CART are obtained by 10-fold cross-validation, see [2].

Experiment II consists of the following steps:

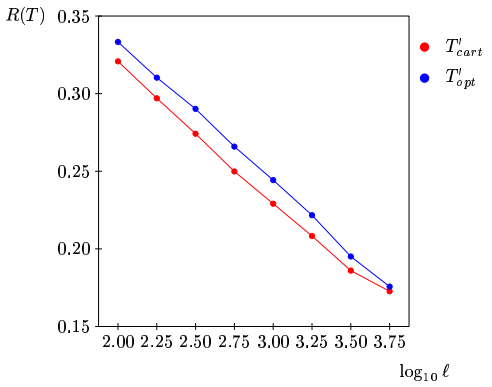

Figure 3: Experiment II with majority ($g = g_2$), attribute noise level $\nu = 0.1$. Mean true errors of 100 trees grown by Algorithm 2 and by CART for different sizes ℓ of training data sets.

1. Use CART with cross-validation to construct tree T'_{cart} on the basis of the sample $\mathcal{L}_g$.
2. Construct the optimally trained tree T'_{opt} on $\mathcal{L}_g$ of the same size as T'_{cart}.
3. Determine the errors of T'_{opt} and T'_{cart} on the distribution D.

The errors of T'_{opt} and T'_{cart} on D are random variables depending on $\mathcal{L}_g$. As in Experiment I, we collected 100 independent realizations of each of these random variates, repeating the procedure 100 times. We, again, estimated the expected errors of T'_{opt} and T'_{cart}, calculated standard errors of the estimates, and tested a possible difference between the expected errors using the two sided paired Student's t-test. We performed the above experiment several times for different values of ℓ, using independent samples for each ℓ. The sizes and data sets used for Experiments II were the same as those used for the Experiments I for the same distribution.

The examples described in the following paragraphs, where T'_{opt} is worse than T'_{cart}, demonstrate situations where the minimization of error on training data for a given size of the tree does not improve the generalization.

5.3.1 Results for majority

The same values of ν, ℓ and M are used as in the first experiment with g_2. By construction, T'_{opt} has smaller error on training data than T'_{cart}. However, as demonstrated by Fig. 3, the error of T'_{opt} on D is larger on average than that of T'_{cart} for all sizes of the data sets. The numerical results are summarized in the tables in Appendix A.8.

5.3.2 Results for two thresholds

The same values of ν, ℓ and M are used as in the first experiment with g_3.

For the smaller noise level, i.e. $\nu = 0.02$, Algorithm 2 was worse than CART, as regards generalization, at data sizes ranging from 316 to 5623, while for $\nu = 0.1$ at all eight sizes.

The results are summarized in Appendices A.9 and A.10.

5.3.3 Results for two squares

The same values of ν, ℓ and M are used as in the first experiment with g_4.

For the distribution without any attribute noise, i.e. $\nu = 0$, the optimally trained trees had, on average, worse generalization than those from CART, for the data sizes from 100 up to 1000. (Above this range of ℓ, the classification task appeared easy for both methods, and the results were comparable.) For the both positive noise levels, i.e. $\nu = 0.02$ and 0.1, Algorithm 2 was worse than CART in the given sense for all the eight data sizes.

The results are summarized in Appendices A.11 to A.13.

5.4 Experiment III

The disadvantage of experiment II is that the procedure of the selection of T'_{opt} was forced to use the size of T'_{cart}, without any opportunity of comparing estimates of true errors of trees of different sizes. Hence, the comparison may not be really fair. In order to avoid this problem, we designed also another setup, where optimization method is allowed to use the information on the exact distribution D to select the size of the resulting tree. This, of course, is not a model of a real life learning method, but it demonstrates that the deficiency of tree optimization for learning some structures comes already from the tree construction part, and not from the size selection.

Experiment III consists of the following steps:

1. Use CART with cross-validation to construct tree T'_{cart} on the basis of the sample $\mathcal{L}_g$.
2. Determine the error of T'_{cart} on the distribution D.
3. Construct the optimally trained trees for all sizes up to M and select T''_{opt}, which has the smallest error on D.

The errors of T''_{opt} and T'_{cart} on D are random variables depending on $\mathcal{L}_g$. As in Experiments I and II, we collected 100 independent realizations of each of these random variates, repeating the procedure 100 times for each size ℓ of $\mathcal{L}_g$, using independent samples for each ℓ. We, again, estimated the expected errors of T''_{opt} and T'_{cart}, calculated standard errors of the estimates, and tested a possible difference between the expected errors using the two sided paired Student's t-test. The sizes and data sets used for Experiments III were the same as those used for the Experiments I and II for the same distribution.

Note that the set of optimally trained trees, from which T''_{opt} is selected, contains also trees with the minimum size among trees with error not larger than the error of T'_{cart}. Hence, the examples described in the following paragraphs, where T''_{opt} is worse than T'_{cart}, demonstrate situations where the minimization of size for a given bound on the error does not improve the generalization.

5.4.1 Results for majority

Trees T''_{opt} are worse than T'_{cart} for sizes $\ell = 10^3, 10^{3.25}$ and $10^{3.5}$, i.e. $\ell = 1000$, 1778, and 3162, of $|L_g|$. The differences in favor of CART at the sizes cited, as displayed at Fig. 4, do not seem to be convincing, but the tables in Appendix A.14 demonstrate high levels of statistical significance.

5.4.2 Results for two thresholds

The same values of ν, ℓ and M are used as in the Experiments I and II with g_3.

For the noise level $\nu = 0.02$, Algorithm 2 was worse than CART, as regards generalization, at data sizes 1000 and 1778, while for $\nu = 0.1$ at sizes 3162 and 5623.

The results are summarized in Appendices A.15 and A.16.

5.4.3 Results for two squares

The same values of ν, ℓ and M are used as in the Experiments I and II with g_4.

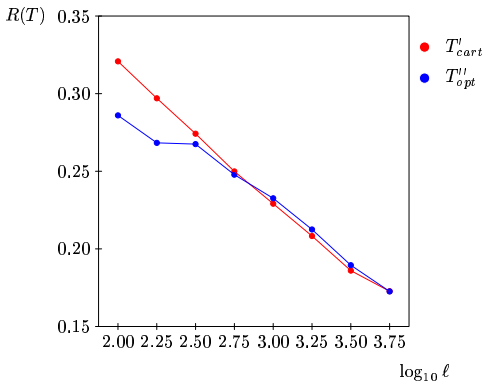

Figure 4: Experiment III with majority ($g = g_2$), attribute noise level $\nu = 0.1$. Mean true errors of 100 trees grown by Algorithm 2 and by CART for different sizes ℓ of training data sets.

For the distribution without any attribute noise, i.e. $\nu = 0$, the phenomenon of the inferiority of the optimally trained trees in comparison with those from CART was not very marked. Only for one size, $\ell = 562$, the comparison of the two methods yielded a significant result at the 5% level ($p = 0.018$), and for another size, $\ell = 1000$, the p-value of the t-test was slightly above 5% ($p = 0.073$). For $\nu = 0.02$, Algorithm 2 was less successful than CART, as regards generalization, at sizes from 562 to 1778 (for $\ell = 562$, the p-values was 0.033, while for the other sizes much higher levels of statistical significance were attained). For $\nu = 0.1$, the range of sizes where Algorithm 2 was beaten by CART was from 1778 to 5623.

The results are summarized in Appendices A.17 to A.19.

6 Conclusion

We have shown by the parity example that there are problems where elements of optimization, if made computationally feasible, may considerably help to achieve good generalization in comparison with the traditional tree-based methods.

We have, however, also shown clearly (using statistical tests of significance) that for a kind of problems optimization yields worse results than the methods based on recursive partitioning. Moreover, by analysis of these problems, we derived new examples of situations, where the preference of small models suggested by Occam's razor principle does not improve the quality of the model.

Further investigation of these phenomena is needed in order to understand better the conditions, under which the optimization helps. This can lead to improvements of the known tree construction techniques by using elements of optimization where appropriate.

References

[1] Blumer A., Ehrenfeucht A., Haussler D. and Warmuth M. (1987). Occam's Razor. *Information Processing Letters* 24, 377–380.

[2] Breiman L., Friedman J.H., Olshen R.A. and Stone C.J. (1984). *Classification and Regression Trees*. Wadsworth, Belmont CA.

[3] Domingos P. (1999). The Role of Occam's Razor in Knowledge Discovery. *Data Mining and Knowledge Discovery* 3 (4), 409-425.

[4] Elomaa T. (1996). *Tools and Techniques for Decision Tree Learning.* Ph. D. Thesis, http://www.cs.helsinki.fi/TR/A-1996/2/.

[5] Impagliazzo R., personal communication.

[6] Quinlan J.R. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufman Publishers, San Mateo.

[7] Loh W.-Y. and Shih Y.-S. (1997). Split selection methods for classification trees. *Statistica Sinica* 7, 815–840.

[8] Murphy P.M. and Pazzani M.J. (1994). Exploring the Decision Forest: An Empirical Investigation of Occam's Razor in Decision Tree Induction. *J. of Art. Int. Res.* 1, 257–275.

[9] Savický P., Klaschka J. and Antoch J. (2000). Optimal Classification Trees. In: *COMPSTAT 2000, Proceedings in Computational Statistics* (ed. J.G. Bethlehem and P.G.M. vad der Heijden), 427–432. Physica-Verlag, Heidelberg.

[10] Webb I.G. (1996). Further Experimental Evidence against the Utility of Occam's Razor. *J. of Art. Int. Res.* 4, 397–417.

[11] L'Ecuyer P. (1999). Good Parameter Sets for Combined Multiple Recursive Random Number Generators. *Operations Research* 47 (1), 159–164.

[12] L'Ecuyer P., source code, http://www.iro.umontreal.ca/~lecuyer/myftp/papers/combmrg2.c

A Appendix

A.1 Experiment I, parity, noise 2%

Tree sizes:

data files size	Tree size (mean ± SEM)					
	Optimization			CART		
100	73.4	±	1.7	22.2	±	1.9
178	47.1	±	2.2	50.2	±	3.2
316	33.3	±	0.3	116.2	±	4.4
562	32.1	±	0.1	212.5	±	3.8
1000	32.3	±	0.1	300.0	±	7.3

Error on training data:

data files size	Error (%) on training data (mean ± SEM)					
	Optimization			CART		
100	0.76	±	0.11	20.28	±	1.70
178	4.37	±	0.30	14.63	±	1.52
316	8.25	±	0.17	7.50	±	1.00
562	9.23	±	0.12	5.07	±	0.20
1000	9.37	±	0.10	6.62	±	0.17

Error on validation data:

data files size	Error (%) on validation data (mean ± SEM)					
	Optimization			CART		
100	18.71	±	0.72	41.72	±	0.59
178	17.58	±	0.47	40.19	±	0.55
316	11.18	±	0.34	36.80	±	0.54
562	9.23	±	0.19	32.06	±	0.50
1000	9.12	±	0.12	24.97	±	0.42

True error:

data files size	True error in % (mean ± SEM)						p-value (paired t)
	Optimization			CART			
100	21.68	±	0.54	47.86	±	0.36	0.000000
178	20.68	±	0.38	44.21	±	0.53	0.000000
316	12.06	±	0.28	39.03	±	0.53	0.000000
562	9.33	±	0.04	32.15	±	0.41	0.000000
1000	9.26	±	0.01	25.24	±	0.33	0.000000

A.2 Experiment I, majority, noise 10%

Tree sizes:

data files size	Tree size (mean ± SEM)	
	Optimization	CART
100	35.6 ± 2.9	17.4 ± 1.1
178	60.0 ± 5.0	24.3 ± 1.6
316	42.1 ± 3.9	38.2 ± 2.7
562	40.6 ± 2.6	53.6 ± 3.4
1000	56.2 ± 3.0	71.0 ± 3.2
1778	71.1 ± 2.6	78.0 ± 2.5
3162	73.7 ± 1.4	77.6 ± 1.8
5623	70.0 ± 0.5	72.0 ± 0.5

Error on training data:

data files size	Error (%) on training data (mean ± SEM)	
	Optimization	CART
100	6.49 ± 0.70	10.82 ± 0.86
178	6.64 ± 0.64	12.11 ± 0.76
316	11.17 ± 0.62	12.72 ± 0.60
562	13.32 ± 0.44	12.95 ± 0.41
1000	14.26 ± 0.33	13.93 ± 0.24
1778	14.90 ± 0.16	15.14 ± 0.14
3162	16.06 ± 0.08	16.20 ± 0.08
5623	16.70 ± 0.05	16.68 ± 0.05

Error on validation data:

data files size	Error (%) on validation data (mean ± SEM)	
	Optimization	CART
100	23.55 ± 0.53	26.22 ± 0.61
178	24.11 ± 0.41	26.43 ± 0.45
316	24.05 ± 0.31	24.47 ± 0.36
562	23.39 ± 0.25	23.25 ± 0.28
1000	22.25 ± 0.16	21.74 ± 0.20
1778	20.74 ± 0.14	19.90 ± 0.13
3162	18.79 ± 0.10	18.19 ± 0.10
5623	17.18 ± 0.08	17.04 ± 0.08

True error:

data files size	True error in % (mean ± SEM)		p-value (paired t)
	Optimization	CART	
100	30.96 ± 0.20	30.72 ± 0.27	0.423116
178	29.02 ± 0.19	29.00 ± 0.23	0.931057
316	28.31 ± 0.18	27.13 ± 0.15	0.000000
562	25.89 ± 0.13	24.60 ± 0.11	0.000000
1000	24.02 ± 0.10	22.66 ± 0.11	0.000000
1778	21.71 ± 0.08	20.51 ± 0.08	0.000000
3162	19.17 ± 0.06	18.43 ± 0.07	0.000000
5623	17.31 ± 0.04	17.16 ± 0.03	0.000000

A.3 Experiment I, function g_3 (two thresholds), 2% noise

Tree sizes:

data files size	Tree size (mean ± SEM)	
	Optimization	CART
100	34.8 ± 2.5	20.2 ± 0.7
178	45.9 ± 3.7	28.8 ± 1.1
316	39.8 ± 2.6	40.0 ± 1.4
562	43.6 ± 1.2	52.5 ± 1.1
1000	52.6 ± 1.2	59.5 ± 0.9
1778	60.9 ± 0.7	61.5 ± 0.4
3162	61.4 ± 0.1	61.1 ± 0.1
5623	61.0 ± 0.0	61.0 ± 0.0

Error on training data:

data files size	Error (%) on training data (mean ± SEM)	
	Optimization	CART
100	3.00 ± 0.45	4.49 ± 0.51
178	3.14 ± 0.30	4.84 ± 0.42
316	3.63 ± 0.25	4.13 ± 0.28
562	3.58 ± 0.16	3.69 ± 0.14
1000	3.78 ± 0.09	3.83 ± 0.08
1778	3.83 ± 0.05	3.94 ± 0.05
3162	3.92 ± 0.04	3.94 ± 0.04
5623	3.95 ± 0.03	3.95 ± 0.03

Error on validation data:

data files size	Error (%) on validation data (mean ± SEM)	
	Optimization	CART
100	14.85 ± 0.50	19.35 ± 0.53
178	12.38 ± 0.32	14.94 ± 0.47
316	10.45 ± 0.28	11.22 ± 0.31
562	7.78 ± 0.17	7.56 ± 0.19
1000	6.24 ± 0.10	5.45 ± 0.11
1778	4.71 ± 0.08	4.28 ± 0.06
3162	4.03 ± 0.05	4.01 ± 0.05
5623	4.00 ± 0.04	4.01 ± 0.04

True error:

data files size	True error in % (mean ± SEM)		p-value (paired t)
	Optimization	CART	
100	21.71 ± 0.33	22.64 ± 0.34	0.032658
178	16.50 ± 0.27	17.36 ± 0.32	0.016207
316	12.35 ± 0.18	12.23 ± 0.19	0.601467
562	9.22 ± 0.10	8.35 ± 0.11	0.000000
1000	6.88 ± 0.06	5.78 ± 0.06	0.000000
1778	5.03 ± 0.04	4.43 ± 0.03	0.000000
3162	4.04 ± 0.01	3.99 ± 0.01	0.000972
5623	3.96 ± 0.00	3.96 ± 0.00	1.000000

A.4 Experiment I with g_3 (two thresholds), 10% attribute noise

Tree sizes:

data files size	Tree size (mean ± SEM)	
	Optimization	CART
100	30.6 ± 2.8	15.6 ± 1.0
178	46.8 ± 4.7	25.9 ± 1.7
316	37.8 ± 3.2	32.1 ± 2.1
562	30.0 ± 1.3	39.6 ± 2.1
1000	37.6 ± 1.7	48.9 ± 2.1
1778	46.0 ± 1.4	55.2 ± 1.3
3162	53.2 ± 1.3	61.1 ± 1.3
5623	60.7 ± 0.8	63.1 ± 0.7

Error on training data:

data files size	Error (%) on training data (mean ± SEM)	
	Optimization	CART
100	8.46 ± 0.84	13.15 ± 0.87
178	9.43 ± 0.82	12.95 ± 0.77
316	11.63 ± 0.55	14.56 ± 0.55
562	15.18 ± 0.31	15.16 ± 0.32
1000	15.99 ± 0.22	15.99 ± 0.21
1778	16.24 ± 0.14	16.33 ± 0.12
3162	16.71 ± 0.09	16.75 ± 0.08
5623	16.93 ± 0.06	16.98 ± 0.05

Error on validation data:

data files size	Error (%) on validation data (mean ± SEM)	
	Optimization	CART
100	24.53 ± 0.48	27.93 ± 0.57
178	24.88 ± 0.40	27.36 ± 0.47
316	24.62 ± 0.29	25.85 ± 0.34
562	22.45 ± 0.26	22.88 ± 0.27
1000	20.91 ± 0.16	20.95 ± 0.18
1778	19.64 ± 0.12	19.35 ± 0.13
3162	18.47 ± 0.10	17.96 ± 0.10
5623	17.64 ± 0.06	17.32 ± 0.07

True error:

data files size	True error in % (mean ± SEM)		p-value (paired t)
	Optimization	CART	
100	32.95 ± 0.27	33.11 ± 0.28	0.621602
178	31.09 ± 0.21	30.65 ± 0.24	0.114377
316	28.30 ± 0.20	27.66 ± 0.18	0.005723
562	24.81 ± 0.15	24.35 ± 0.13	0.001027
1000	22.19 ± 0.09	21.74 ± 0.09	0.000457
1778	20.34 ± 0.06	19.67 ± 0.06	0.000000
3162	18.87 ± 0.04	18.22 ± 0.04	0.000000
5623	17.82 ± 0.02	17.46 ± 0.03	0.000000

A.5 Experiment I with two squares without attribute noise

Tree sizes:

data files size	Tree size (mean ± SEM)	
	Optimization	CART
100	36.4 ± 2.4	20.6 ± 0.7
178	62.6 ± 3.6	32.3 ± 0.9
316	65.5 ± 3.5	52.7 ± 1.1
562	67.1 ± 1.6	73.3 ± 0.8
1000	81.4 ± 0.5	84.7 ± 0.4
1778	87.7 ± 0.1	88.2 ± 0.1
3162	88.0 ± 0.0	88.3 ± 0.1
5623	88.0 ± 0.0	88.2 ± 0.0

Error on training data:

data files size	Error (%) on training data (mean ± SEM)	
	Optimization	CART
100	2.49 ± 0.48	3.00 ± 0.41
178	1.03 ± 0.25	2.27 ± 0.27
316	0.84 ± 0.14	0.96 ± 0.19
562	0.35 ± 0.06	0.26 ± 0.04
1000	0.06 ± 0.01	0.06 ± 0.01
1778	0.00 ± 0.00	0.01 ± 0.00
3162	0.00 ± 0.00	0.00 ± 0.00
5623	0.00 ± 0.00	0.00 ± 0.00

Error on validation data:

data files size	Error (%) on validation data (mean ± SEM)	
	Optimization	CART
100	14.14 ± 0.39	18.03 ± 0.56
178	11.04 ± 0.32	14.13 ± 0.45
316	8.03 ± 0.26	8.72 ± 0.29
562	4.48 ± 0.14	4.19 ± 0.16
1000	1.10 ± 0.07	0.95 ± 0.08
1778	0.06 ± 0.02	0.06 ± 0.02
3162	0.00 ± 0.00	0.00 ± 0.00
5623	0.00 ± 0.00	0.00 ± 0.00

True error:

data files size	True error in % (mean ± SEM)		p-value (paired t)
	Optimization	CART	
100	20.80 ± 0.34	21.18 ± 0.34	0.394557
178	14.40 ± 0.25	15.88 ± 0.26	0.000007
316	10.03 ± 0.19	9.89 ± 0.20	0.553002
562	5.30 ± 0.13	4.41 ± 0.13	0.000000
1000	1.32 ± 0.07	1.00 ± 0.06	0.000035
1778	0.06 ± 0.01	0.08 ± 0.02	0.236657
3162	0.00 ± 0.00	0.00 ± 0.00	0.319749
5623	0.00 ± 0.00	0.00 ± 0.00	1.000000

A.6 Experiment I with two squares and 2% attribute noise

Tree sizes:

data files size	Tree size (mean ± SEM)	
	Optimization	CART
100	42.5 ± 2.7	18.5 ± 0.8
178	56.2 ± 3.6	31.6 ± 1.2
316	53.8 ± 2.9	53.4 ± 1.6
562	59.1 ± 1.9	73.0 ± 2.0
1000	75.7 ± 1.6	86.0 ± 1.7
1778	86.6 ± 1.0	87.7 ± 0.8
3162	88.1 ± 0.3	88.7 ± 0.2
5623	88.1 ± 0.1	88.5 ± 0.1

Error on training data:

data files size	Error (%) on training data (mean ± SEM)	
	Optimization	CART
100	2.54 ± 0.42	6.25 ± 0.60
178	2.82 ± 0.40	4.50 ± 0.40
316	3.11 ± 0.28	3.29 ± 0.29
562	3.59 ± 0.22	3.39 ± 0.18
1000	3.61 ± 0.10	3.65 ± 0.09
1778	3.96 ± 0.05	4.10 ± 0.06
3162	4.17 ± 0.04	4.18 ± 0.04
5623	4.13 ± 0.02	4.13 ± 0.02

Error on validation data:

data files size	Error (%) on validation data (mean ± SEM)	
	Optimization	CART
100	17.37 ± 0.46	20.73 ± 0.56
178	14.81 ± 0.36	17.88 ± 0.46
316	12.80 ± 0.24	13.44 ± 0.32
562	10.44 ± 0.17	10.38 ± 0.19
1000	7.81 ± 0.10	7.26 ± 0.13
1778	5.59 ± 0.10	5.17 ± 0.10
3162	4.37 ± 0.05	4.33 ± 0.05
5623	4.17 ± 0.04	4.18 ± 0.04

True error:

data files size	True error in % (mean ± SEM)		p-value (paired t)
	Optimization	CART	
100	23.90 ± 0.30	24.83 ± 0.31	0.017956
178	19.72 ± 0.24	20.15 ± 0.24	0.186803
316	15.88 ± 0.18	15.13 ± 0.19	0.000945
562	12.24 ± 0.13	10.95 ± 0.14	0.000000
1000	8.62 ± 0.07	7.58 ± 0.08	0.000000
1778	5.83 ± 0.06	5.31 ± 0.05	0.000000
3162	4.37 ± 0.02	4.31 ± 0.02	0.006660
5623	4.18 ± 0.00	4.18 ± 0.00	0.740706

A.7 Experiment I with two squares and 10% attribute noise

Tree sizes:

data files size	Tree size (mean ± SEM)	
	Optimization	CART
100	30.9 ± 2.6	14.8 ± 1.0
178	32.7 ± 3.1	23.2 ± 1.5
316	31.9 ± 2.1	37.3 ± 2.5
562	34.7 ± 2.0	52.8 ± 3.1
1000	47.3 ± 2.2	65.4 ± 3.4
1778	59.8 ± 2.3	78.6 ± 2.9
3162	68.5 ± 1.8	82.0 ± 1.7
5623	84.0 ± 1.2	87.6 ± 1.2

Error on training data:

data files size	Error (%) on training data (mean ± SEM)	
	Optimization	CART
100	7.72 ± 0.84	13.87 ± 0.90
178	10.51 ± 0.72	13.34 ± 0.64
316	12.82 ± 0.60	13.65 ± 0.60
562	15.11 ± 0.43	14.15 ± 0.39
1000	15.53 ± 0.29	15.30 ± 0.28
1778	16.31 ± 0.18	16.00 ± 0.14
3162	17.02 ± 0.11	16.88 ± 0.09
5623	17.07 ± 0.06	17.14 ± 0.05

Error on validation data:

data files size	Error (%) on validation data (mean ± SEM)	
	Optimization	CART
100	24.56 ± 0.49	28.20 ± 0.57
178	24.56 ± 0.38	26.77 ± 0.46
316	24.80 ± 0.33	25.74 ± 0.37
562	24.19 ± 0.23	24.42 ± 0.25
1000	22.65 ± 0.18	22.69 ± 0.18
1778	21.19 ± 0.15	20.71 ± 0.15
3162	19.85 ± 0.11	19.43 ± 0.12
5623	18.61 ± 0.07	18.24 ± 0.07

True error:

data files size	True error in % (mean ± SEM)		p-value (paired t)
	Optimization	CART	
100	33.47 ± 0.23	33.21 ± 0.25	0.360935
178	31.18 ± 0.16	30.50 ± 0.21	0.003009
316	29.22 ± 0.19	28.06 ± 0.19	0.000000
562	26.84 ± 0.13	25.85 ± 0.13	0.000000
1000	24.21 ± 0.10	23.71 ± 0.11	0.000030
1778	22.23 ± 0.07	21.30 ± 0.08	0.000000
3162	20.38 ± 0.04	19.65 ± 0.05	0.000000
5623	18.79 ± 0.03	18.34 ± 0.03	0.000000

A.8 Experiment II with majority, noise 10%

Tree sizes:

data files size	Tree size (mean ± SEM)	
	Optimization	CART
100	13.9 ± 1.1	13.9 ± 1.1
178	23.1 ± 1.6	23.1 ± 1.6
316	32.4 ± 2.3	32.4 ± 2.3
562	45.7 ± 3.0	45.7 ± 3.0
1000	63.4 ± 2.9	63.4 ± 2.9
1778	73.9 ± 2.4	74.1 ± 2.5
3162	76.4 ± 1.3	76.4 ± 1.3
5623	72.3 ± 0.6	72.3 ± 0.6

Error on training data:

data files size	Error (%) on training data (mean ± SEM)	
	Optimization	CART
100	12.19 ± 1.02	13.50 ± 0.95
178	10.95 ± 0.84	12.35 ± 0.79
316	11.77 ± 0.57	13.42 ± 0.53
562	12.48 ± 0.45	13.86 ± 0.41
1000	13.22 ± 0.29	14.40 ± 0.26
1778	14.61 ± 0.16	15.27 ± 0.15
3162	15.91 ± 0.07	16.17 ± 0.07
5623	16.61 ± 0.05	16.65 ± 0.05

True error:

data files size	True error in % (mean ± SEM)		p-value (paired t)
	Optimization	CART	
100	33.33 ± 0.30	32.08 ± 0.35	0.000000
178	31.03 ± 0.24	29.70 ± 0.27	0.000000
316	29.02 ± 0.17	27.42 ± 0.17	0.000000
562	26.59 ± 0.15	24.99 ± 0.12	0.000000
1000	24.43 ± 0.11	22.91 ± 0.09	0.000000
1778	22.17 ± 0.07	20.83 ± 0.08	0.000000
3162	19.51 ± 0.07	18.60 ± 0.06	0.000000
5623	17.57 ± 0.05	17.27 ± 0.04	0.000000

A.9 Experiment II, function g_3 (two thresholds), 2% noise

Tree sizes:

data files size	Tree size (mean ± SEM)	
	Optimization	CART
100	17.9 ± 0.9	17.9 ± 0.9
178	29.8 ± 1.0	29.8 ± 1.0
316	41.6 ± 1.2	41.6 ± 1.2
562	53.9 ± 1.3	53.9 ± 1.3
1000	58.3 ± 0.8	58.3 ± 0.8
1778	61.1 ± 0.4	61.1 ± 0.4
3162	61.2 ± 0.1	61.2 ± 0.1
5623	61.0 ± 0.0	61.0 ± 0.0

Error on training data:

data files size	Error (%) on training data (mean ± SEM)	
	Optimization	CART
100	4.83 ± 0.59	5.92 ± 0.60
178	2.53 ± 0.26	4.09 ± 0.31
316	2.25 ± 0.19	3.54 ± 0.23
562	2.61 ± 0.12	3.54 ± 0.15
1000	3.33 ± 0.07	3.85 ± 0.07
1778	3.80 ± 0.05	3.94 ± 0.05
3162	3.92 ± 0.04	3.93 ± 0.04
5623	3.95 ± 0.03	3.95 ± 0.03

True error:

data files size	True error in % (mean ± SEM)		p-value (paired t)
	Optimization	CART	
100	24.42 ± 0.36	23.91 ± 0.36	0.152718
178	18.13 ± 0.29	17.75 ± 0.30	0.239262
316	13.52 ± 0.20	12.55 ± 0.19	0.000058
562	10.15 ± 0.13	8.77 ± 0.14	0.000000
1000	7.38 ± 0.09	6.03 ± 0.07	0.000000
1778	5.31 ± 0.05	4.58 ± 0.04	0.000000
3162	4.13 ± 0.03	4.03 ± 0.01	0.000102
5623	3.96 ± 0.00	3.96 ± 0.00	1.000000

A.10 Experiment II with g_3 (two thresholds), 10% attribute noise

Tree sizes:

data files size	Tree size (mean ± SEM)	
	Optimization	CART
100	14.1 ± 1.1	14.1 ± 1.1
178	23.0 ± 1.8	23.0 ± 1.8
316	34.2 ± 2.2	34.2 ± 2.2
562	37.7 ± 1.8	37.7 ± 1.8
1000	43.7 ± 1.6	43.7 ± 1.6
1778	51.9 ± 1.1	51.9 ± 1.1
3162	60.4 ± 1.1	60.4 ± 1.1
5623	62.5 ± 0.6	62.5 ± 0.6

Error on training data:

data files size	Error (%) on training data (mean ± SEM)	
	Optimization	CART
100	12.91 ± 1.01	14.44 ± 0.93
178	12.16 ± 0.82	13.78 ± 0.78
316	11.55 ± 0.54	13.78 ± 0.52
562	13.69 ± 0.37	15.27 ± 0.33
1000	15.09 ± 0.23	16.30 ± 0.20
1778	15.68 ± 0.11	16.49 ± 0.12
3162	16.32 ± 0.07	16.73 ± 0.07
5623	16.86 ± 0.05	16.98 ± 0.05

True error:

data files size	True error in % (mean ± SEM)		p-value (paired t)
	Optimization	CART	
100	34.87 ± 0.33	33.94 ± 0.30	0.000675
178	32.64 ± 0.27	31.37 ± 0.28	0.000000
316	29.01 ± 0.19	28.25 ± 0.18	0.000070
562	25.66 ± 0.16	24.85 ± 0.16	0.000000
1000	22.84 ± 0.11	22.06 ± 0.10	0.000000
1778	20.67 ± 0.07	19.91 ± 0.07	0.000000
3162	19.19 ± 0.05	18.39 ± 0.04	0.000000
5623	18.02 ± 0.03	17.55 ± 0.03	0.000000

A.11 Experiment II with two squares without attribute noise

Tree sizes:

data files size	Tree size (mean ± SEM)	
	Optimization	CART
100	18.4 ± 0.8	18.4 ± 0.8
178	34.5 ± 0.8	34.5 ± 0.8
316	50.0 ± 1.1	50.0 ± 1.1
562	73.0 ± 0.8	73.0 ± 0.8
1000	84.9 ± 0.3	84.9 ± 0.3
1778	88.0 ± 0.1	88.0 ± 0.1
3162	88.3 ± 0.1	88.3 ± 0.1
5623	88.2 ± 0.0	88.2 ± 0.0

Error on training data:

data files size	Error (%) on training data (mean ± SEM)	
	Optimization	CART
100	3.41 ± 0.55	4.47 ± 0.56
178	0.76 ± 0.20	1.53 ± 0.24
316	0.56 ± 0.12	1.18 ± 0.16
562	0.06 ± 0.03	0.23 ± 0.04
1000	0.01 ± 0.00	0.04 ± 0.01
1778	0.01 ± 0.00	0.00 ± 0.00
3162	0.00 ± 0.00	0.00 ± 0.00
5623	0.00 ± 0.00	0.00 ± 0.00

True error:

data files size	True error in % (mean ± SEM)		p-value (paired t)
	Optimization	CART	
100	23.50 ± 0.38	22.50 ± 0.43	0.001900
178	16.89 ± 0.26	16.01 ± 0.26	0.001433
316	11.60 ± 0.18	10.38 ± 0.21	0.000000
562	5.47 ± 0.14	4.51 ± 0.13	0.000000
1000	1.36 ± 0.07	1.05 ± 0.06	0.000002
1778	0.08 ± 0.02	0.09 ± 0.02	0.285809
3162	0.00 ± 0.00	0.01 ± 0.00	0.319749
5623	0.00 ± 0.00	0.00 ± 0.00	1.000000

A.12 Experiment II with two squares and 2% attribute noise

Tree sizes:

data files size	Tree size (mean ± SEM)	
	Optimization	CART
100	19.4 ± 0.9	19.4 ± 0.9
178	33.2 ± 1.2	33.2 ± 1.2
316	50.2 ± 1.6	50.2 ± 1.6
562	74.8 ± 1.8	74.8 ± 1.8
1000	82.5 ± 1.4	82.7 ± 1.5
1778	86.3 ± 0.8	86.3 ± 0.8
3162	88.5 ± 0.4	88.5 ± 0.4
5623	88.4 ± 0.1	88.4 ± 0.1

Error on training data:

data files size	Error (%) on training data (mean ± SEM)	
	Optimization	CART
100	4.61 ± 0.65	5.66 ± 0.67
178	2.64 ± 0.36	3.78 ± 0.40
316	2.47 ± 0.22	3.50 ± 0.25
562	2.38 ± 0.12	3.07 ± 0.15
1000	3.17 ± 0.08	3.72 ± 0.09
1778	3.95 ± 0.05	4.11 ± 0.05
3162	4.15 ± 0.04	4.17 ± 0.04
5623	4.13 ± 0.02	4.13 ± 0.02

True error:

data files size	True error in % (mean ± SEM)		p-value (paired t)
	Optimization	CART	
100	26.63 ± 0.32	25.43 ± 0.34	0.000017
178	21.48 ± 0.23	20.45 ± 0.27	0.000143
316	17.22 ± 0.19	15.47 ± 0.17	0.000000
562	12.94 ± 0.14	11.19 ± 0.13	0.000000
1000	9.18 ± 0.08	7.81 ± 0.09	0.000000
1778	6.22 ± 0.07	5.51 ± 0.06	0.000000
3162	4.58 ± 0.03	4.44 ± 0.03	0.000000
5623	4.20 ± 0.01	4.19 ± 0.01	0.009907

A.13 Experiment II with two squares and 10% attribute noise

Tree sizes:

data files size	Tree size (mean ± SEM)	
	Optimization	CART
100	14.7 ± 1.0	14.7 ± 1.0
178	26.9 ± 1.7	26.9 ± 1.7
316	39.9 ± 2.7	39.9 ± 2.7
562	45.1 ± 2.6	46.1 ± 3.0
1000	58.3 ± 2.5	58.9 ± 2.7
1778	67.9 ± 2.0	67.9 ± 2.0
3162	77.6 ± 1.7	77.9 ± 1.7
5623	84.7 ± 1.3	84.8 ± 1.3

Error on training data:

data files size	Error (%) on training data (mean ± SEM)	
	Optimization	CART
100	11.70 ± 1.05	13.36 ± 0.97
178	9.80 ± 0.72	11.67 ± 0.70
316	10.96 ± 0.59	12.82 ± 0.59
562	13.05 ± 0.43	14.79 ± 0.39
1000	14.26 ± 0.31	15.61 ± 0.29
1778	15.56 ± 0.14	16.40 ± 0.14
3162	16.54 ± 0.10	16.91 ± 0.09
5623	17.03 ± 0.06	17.14 ± 0.06

True error:

data files size	True error in % (mean ± SEM)		p-value (paired t)
	Optimization	CART	
100	35.01 ± 0.30	34.12 ± 0.30	0.000218
178	31.98 ± 0.20	31.04 ± 0.20	0.000002
316	29.89 ± 0.21	28.48 ± 0.20	0.000000
562	27.67 ± 0.15	26.38 ± 0.14	0.000000
1000	24.92 ± 0.11	23.98 ± 0.11	0.000000
1778	22.64 ± 0.07	21.53 ± 0.07	0.000000
3162	20.66 ± 0.05	19.89 ± 0.05	0.000000
5623	19.05 ± 0.04	18.59 ± 0.04	0.000000

A.14 Experiment III with majority

Tree sizes:

data files size	Tree size (mean ± SEM)	
	Optimization	CART
100	58.2 ± 3.3	13.9 ± 1.1
178	120.9 ± 5.0	23.1 ± 1.6
316	38.6 ± 3.8	32.4 ± 2.3
562	32.7 ± 1.4	45.7 ± 3.0
1000	44.5 ± 1.6	63.4 ± 2.9
1778	61.6 ± 1.6	74.1 ± 2.5
3162	67.0 ± 0.9	76.4 ± 1.3
5623	69.3 ± 0.4	72.3 ± 0.6

Error on training data:

data files size	Error (%) on training data (mean ± SEM)	
	Optimization	CART
100	4.79 ± 0.66	13.50 ± 0.95
178	3.76 ± 0.47	12.35 ± 0.79
316	11.32 ± 0.45	13.42 ± 0.53
562	14.03 ± 0.28	13.86 ± 0.41
1000	15.05 ± 0.21	14.40 ± 0.26
1778	15.36 ± 0.13	15.27 ± 0.15
3162	16.34 ± 0.07	16.17 ± 0.07
5623	16.72 ± 0.05	16.65 ± 0.05

True error:

data files size	True error in % (mean ± SEM)		p-value (paired t)
	Optimization	CART	
100	28.60 ± 0.16	32.08 ± 0.35	0.000000
178	26.83 ± 0.15	29.70 ± 0.27	0.000000
316	26.75 ± 0.11	27.42 ± 0.17	0.000170
562	24.78 ± 0.09	24.99 ± 0.12	0.094232
1000	23.26 ± 0.07	22.91 ± 0.09	0.000044
1778	21.25 ± 0.07	20.83 ± 0.08	0.000000
3162	18.95 ± 0.05	18.60 ± 0.06	0.000000
5623	17.26 ± 0.03	17.27 ± 0.04	0.633092

A.15 Experiment III with two thresholds and 2% attribute noise

Tree sizes:

data files size	Tree size (mean ± SEM)	
	Optimization	CART
100	49.4 ± 2.4	17.9 ± 0.9
178	58.7 ± 4.4	29.8 ± 1.0
316	55.3 ± 4.7	41.6 ± 1.2
562	45.3 ± 2.1	53.9 ± 1.3
1000	52.6 ± 0.7	58.3 ± 0.8
1778	61.0 ± 0.5	61.1 ± 0.4
3162	61.3 ± 0.1	61.2 ± 0.1
5623	61.0 ± 0.0	61.0 ± 0.0

Error on training data:

data files size	Error (%) on training data (mean ± SEM)	
	Optimization	CART
100	1.03 ± 0.18	5.92 ± 0.60
178	1.89 ± 0.18	4.09 ± 0.31
316	2.92 ± 0.19	3.54 ± 0.23
562	3.48 ± 0.12	3.54 ± 0.15
1000	3.68 ± 0.08	3.85 ± 0.07
1778	3.81 ± 0.05	3.94 ± 0.05
3162	3.92 ± 0.04	3.93 ± 0.04
5623	3.95 ± 0.03	3.95 ± 0.03

True error:

data files size	True error in % (mean ± SEM)		p-value (paired t)
	Optimization	CART	
100	18.84 ± 0.21	23.91 ± 0.36	0.000000
178	14.64 ± 0.19	17.75 ± 0.30	0.000000
316	11.07 ± 0.13	12.55 ± 0.19	0.000000
562	8.61 ± 0.08	8.77 ± 0.14	0.213052
1000	6.54 ± 0.05	6.03 ± 0.07	0.000000
1778	4.91 ± 0.04	4.58 ± 0.04	0.000000
3162	4.02 ± 0.01	4.03 ± 0.01	0.471820
5623	3.96 ± 0.00	3.96 ± 0.00	0.319749

A.16 Experiment III with g_3 (two thresholds), 10% attribute noise

Tree sizes:

data files size	Tree size (mean ± SEM)	
	Optimization	CART
100	35.6 ± 2.9	14.1 ± 1.1
178	48.9 ± 5.1	23.0 ± 1.8
316	25.6 ± 1.5	34.2 ± 2.2
562	28.7 ± 0.8	37.7 ± 1.8
1000	33.5 ± 0.8	43.7 ± 1.6
1778	43.9 ± 0.8	51.9 ± 1.1
3162	52.0 ± 0.8	60.4 ± 1.1
5623	60.8 ± 0.6	62.5 ± 0.6

Error on training data:

data files size	Error (%) on training data (mean ± SEM)	
	Optimization	CART
100	7.28 ± 0.74	14.44 ± 0.93
178	9.25 ± 0.62	13.78 ± 0.78
316	13.33 ± 0.34	13.78 ± 0.52
562	15.20 ± 0.24	15.27 ± 0.33
1000	16.27 ± 0.16	16.30 ± 0.20
1778	16.28 ± 0.11	16.49 ± 0.12
3162	16.71 ± 0.08	16.73 ± 0.07
5623	16.91 ± 0.05	16.98 ± 0.05

True error:

data files size	True error in % (mean ± SEM)		p-value (paired t)
	Optimization	CART	
100	30.48 ± 0.19	33.94 ± 0.30	0.000000
178	29.02 ± 0.14	31.37 ± 0.28	0.000000
316	26.46 ± 0.15	28.25 ± 0.18	0.000000
562	23.76 ± 0.11	24.85 ± 0.16	0.000000
1000	21.52 ± 0.06	22.06 ± 0.10	0.000000
1778	19.96 ± 0.05	19.91 ± 0.07	0.419603
3162	18.69 ± 0.03	18.39 ± 0.04	0.000000
5623	17.76 ± 0.02	17.55 ± 0.03	0.000000

A.17 Experiment III with two squares without attribute noise

Tree sizes:

data files size	Tree size (mean ± SEM)	
	Optimization	CART
100	57.8 ± 2.0	18.4 ± 0.8
178	84.1 ± 3.1	34.5 ± 0.8
316	80.4 ± 3.3	50.0 ± 1.1
562	72.6 ± 1.8	73.0 ± 0.8
1000	82.9 ± 0.7	84.9 ± 0.3
1778	87.7 ± 0.1	88.0 ± 0.1
3162	88.0 ± 0.0	88.3 ± 0.1
5623	88.0 ± 0.0	88.2 ± 0.0

Error on training data:

data files size	Error (%) on training data (mean ± SEM)	
	Optimization	CART
100	0.12 ± 0.05	4.47 ± 0.56
178	0.20 ± 0.07	1.53 ± 0.24
316	0.30 ± 0.06	1.18 ± 0.16
562	0.18 ± 0.04	0.23 ± 0.04
1000	0.04 ± 0.01	0.04 ± 0.01
1778	0.00 ± 0.00	0.00 ± 0.00
3162	0.00 ± 0.00	0.00 ± 0.00
5623	0.00 ± 0.00	0.00 ± 0.00

True error:

data files size	True error in % (mean ± SEM)		p-value (paired t)
	Optimization	CART	
100	17.55 ± 0.17	22.50 ± 0.43	0.000000
178	12.89 ± 0.14	16.01 ± 0.26	0.000000
316	9.10 ± 0.15	10.38 ± 0.21	0.000000
562	4.82 ± 0.10	4.51 ± 0.13	0.017849
1000	1.16 ± 0.06	1.05 ± 0.06	0.073756
1778	0.05 ± 0.01	0.09 ± 0.02	0.017893
3162	0.00 ± 0.00	0.01 ± 0.00	0.179556
5623	0.00 ± 0.00	0.00 ± 0.00	1.000000

A.18 Experiment III with two squares and 2% attribute noise

Tree sizes:

data files size	Tree size (mean ± SEM)	
	Optimization	CART
100	61.0 ± 2.3	19.4 ± 0.9
178	79.7 ± 3.7	33.2 ± 1.2
316	61.3 ± 3.7	50.2 ± 1.6
562	61.2 ± 1.5	74.8 ± 1.8
1000	75.0 ± 1.3	82.7 ± 1.5
1778	86.0 ± 0.8	86.3 ± 0.8
3162	88.2 ± 0.2	88.5 ± 0.4
5623	88.0 ± 0.0	88.4 ± 0.1

Error on training data:

data files size	Error (%) on training data (mean ± SEM)	
	Optimization	CART
100	0.80 ± 0.17	5.66 ± 0.67
178	1.36 ± 0.19	3.78 ± 0.40
316	3.08 ± 0.25	3.50 ± 0.25
562	3.20 ± 0.15	3.07 ± 0.15
1000	3.58 ± 0.10	3.72 ± 0.09
1778	3.97 ± 0.05	4.11 ± 0.05
3162	4.17 ± 0.04	4.17 ± 0.04
5623	4.13 ± 0.02	4.13 ± 0.02

True error:

data files size	True error in % (mean ± SEM)		p-value (paired t)
	Optimization	CART	
100	21.37 ± 0.20	25.43 ± 0.34	0.000000
178	17.56 ± 0.16	20.45 ± 0.27	0.000000
316	14.73 ± 0.14	15.47 ± 0.17	0.000201
562	11.43 ± 0.09	11.19 ± 0.13	0.032535
1000	8.26 ± 0.06	7.81 ± 0.09	0.000001
1778	5.71 ± 0.06	5.51 ± 0.06	0.000015
3162	4.35 ± 0.02	4.44 ± 0.03	0.000099
5623	4.17 ± 0.00	4.19 ± 0.01	0.003282

A.19 Experiment III with two squares and 10% attribute noise

Tree sizes:

data files size	Tree size (mean ± SEM)	
	Optimization	CART
100	40.9 ± 3.1	14.7 ± 1.0
178	39.8 ± 3.7	26.9 ± 1.7
316	27.1 ± 1.2	39.9 ± 2.7
562	31.2 ± 1.2	46.1 ± 3.0
1000	39.1 ± 1.3	58.9 ± 2.7
1778	53.2 ± 1.5	67.9 ± 2.0
3162	71.8 ± 1.5	77.9 ± 1.7
5623	85.8 ± 1.0	84.8 ± 1.3

Error on training data:

data files size	Error (%) on training data (mean ± SEM)	
	Optimization	CART
100	6.37 ± 0.68	13.36 ± 0.97
178	9.26 ± 0.63	11.67 ± 0.70
316	13.01 ± 0.35	12.82 ± 0.59
562	15.28 ± 0.29	14.79 ± 0.39
1000	16.32 ± 0.22	15.61 ± 0.29
1778	16.61 ± 0.14	16.40 ± 0.14
3162	16.81 ± 0.10	16.91 ± 0.09
5623	16.99 ± 0.06	17.14 ± 0.06

True error:

data files size	True error in % (mean ± SEM)		p-value (paired t)
	Optimization	CART	
100	30.99 ± 0.13	34.12 ± 0.30	0.000000
178	29.56 ± 0.12	31.04 ± 0.20	0.000000
316	27.63 ± 0.13	28.48 ± 0.20	0.000000
562	25.73 ± 0.09	26.38 ± 0.14	0.000006
1000	23.47 ± 0.07	23.98 ± 0.11	0.000000
1778	21.79 ± 0.06	21.53 ± 0.07	0.000129
3162	20.14 ± 0.04	19.89 ± 0.05	0.000000
5623	18.70 ± 0.03	18.59 ± 0.04	0.000679