On Sparse Optimal Regression Trees

Rafael Blanquero
Universidad de Sevilla
47 PUBLICATIONS **301** CITATIONS
SEE PROFILE

Emilio Carrizosa
Universidad de Sevilla
202 PUBLICATIONS **1,926** CITATIONS
SEE PROFILE

Cristina Molero-Río
Universidad de Sevilla
7 PUBLICATIONS **8** CITATIONS
SEE PROFILE

Dolores Romero Morales
Copenhagen Business School
68 PUBLICATIONS **990** CITATIONS
SEE PROFILE

On Sparse Optimal Regression Trees

Rafael Blanquero[1], Emilio Carrizosa[1], Cristina Molero-Río[1,*], and Dolores Romero Morales[2]

[1]Instituto de Matemáticas de la Universidad de Sevilla, Seville, Spain

{rblanquero, ecarrizosa, mmolero}@us.es

[2]Copenhagen Business School, Frederiksberg, Denmark

drm.eco@cbs.dk

May 2, 2020

Abstract

In this paper, we model an optimal regression tree through a continuous optimization problem, where a compromise between prediction accuracy and sparsity is sought. Our approach can accommodate other desirable properties for the regression task, such as performance constraints in critical groups. The computational experience reported shows the outperformance of our approach in terms of prediction accuracy against standard benchmark regression methods such as CART, OLS and LASSO. Unlike other more sophisticated tree-based approaches, with excellent prediction performance but a lack of control over sparsity, as is the case for RF, we illustrate our ability to easily trade in some of our prediction accuracy for a gain in sparsity.

Key words Classification and Regression Trees; Optimal Regression Trees; Sparsity; Nonlinear Programming

[*]Corresponding author

1 Introduction

Regression Analysis is one of the most used tasks in Statistics and Machine Learning [24]. The classic linear regression is known to be outperformed by many proposals that apply non-linear techniques, such as tree-based methods, which are the focus of this paper. Tree-based methods [13, 25, 45] are appealing due to their learning performance and, since they are rule-based, seen as interpretable [1, 2, 12, 19, 22, 28, 33, 34, 38, 42].

Building optimal decision trees is an NP-complete task [26]. For this reason, greedy heuristic procedures such as CART [11] have been proposed, yielding suboptimal trees instead. Even though some attempts [3] were made in the past, the latest advances in both computer performance and Mathematical Optimization have led to a growing research on building such optimal decision trees [4, 5, 7, 8, 16, 18, 23, 36, 43, 44].

The modeling of a decision tree via Mathematical Optimization yields, in general, an improvement in prediction accuracy with respect to traditional approaches, but, equally important, it allows the user to easily deal with desirable properties in Machine Learning that globally involve all the decision rules along the tree. Such is the case of sparsity [41]. While heuristic procedures, such as CART, or more sophisticated tree-based approaches, such as Random Forest (RF) [6, 10, 17, 21], find it hard to control the number of predictor variables to be used across the tree [14, 15, 39], optimized approaches are flexible enough to model this objective directly [8]. In this paper, we tackle this issue and propose the Sparse Optimal Randomized Regression Tree (S-ORRT). An S-ORRT seeks a good tradeoff between both prediction accuracy and sparsity, obtained by minimizing the mean squared error over the training sample, as customary in Regression Analysis, plus a regularization term.

Optimal regression trees [4, 16, 43] have been recently formulated using mixed-integer models. These models include an integer decision variable for each individual, as well as one for each predictor variable. The resulting combinatorial framework hinders the tractability of the problem when the dimensionality of the data grows, yielding a significant computational effort even for small data sets. Local-search strategies have been proposed to alleviate the computational burden of these procedures [16] at the expense of controlling sparsity in a weaker way, namely locally, i.e., the predictor variables used in a given node, as opposed to globally, i.e., the predictor variables used across the whole tree. Our approach considers a continuous optimization model instead, where sparsity across the whole tree is controlled. This is achieved through (i) the inclusion of a continuous cumulative density function F at each branch node that smoothens the transition from the left child node to the right one, and (ii) the use of the ℓ_∞-norm to control global sparsity.

The remainder of the paper is organized as follows. In Section 2, we introduce the S-ORRT and its mathematical formulation. Some theoretical properties of S-ORRT are discussed in Section 3. In Section 4, our computational experience is reported. We illustrate that S-ORRT outperforms the benchmark regression methods CART, OLS and LASSO in terms of prediction accuracy. Moreover, we show the ability of S-ORRT to easily trade in some prediction accuracy for a gain in sparsity, unlike more sophisticated tree-based approaches. Finally, conclusions and possible lines of future research are provided in Section 5.

2 Sparse Optimal Randomized Regression Trees

Let $\mathcal{I}$ be a given set of individuals. Each individual $i \in \mathcal{I}$ has associated a pair $(\boldsymbol{x}_i, y_i)$, where $\boldsymbol{x}_i$ represents the p-dimensional vector of predictor variables of individual i, and $y_i \in \mathbb{R}$ indicates the value of the response variable.

A Sparse Optimal Randomized Regression Tree (S-ORRT) is an optimal binary regression tree of a given depth D, obtained by controlling two objectives simultaneously: the mean squared error over the training sample (prediction accuracy), and the number of predictor variables to be used along the whole tree (sparsity). We briefly sketch here this randomized framework. For further details on the construction of optimal randomized trees, the reader is referred to [7, 8]. Figure 1 shows the structure of an S-ORRT of depth $D = 2$. Unlike classic decision trees, oblique cuts, on which more than one predictor variable is involved, are implemented. S-ORRTs are modeled by means of a Non-Linear Continuous Optimization (NLCO) formulation. The usual deterministic yes/no rule at each branch node is replaced by a smoother rule: a probabilistic decision rule at each branch node, induced by a cumulative density function (CDF) F, is obtained. Therefore, the movements in S-ORRTs can be seen as randomized: at a given branch node of a S-ORRT, a random variable will be generated

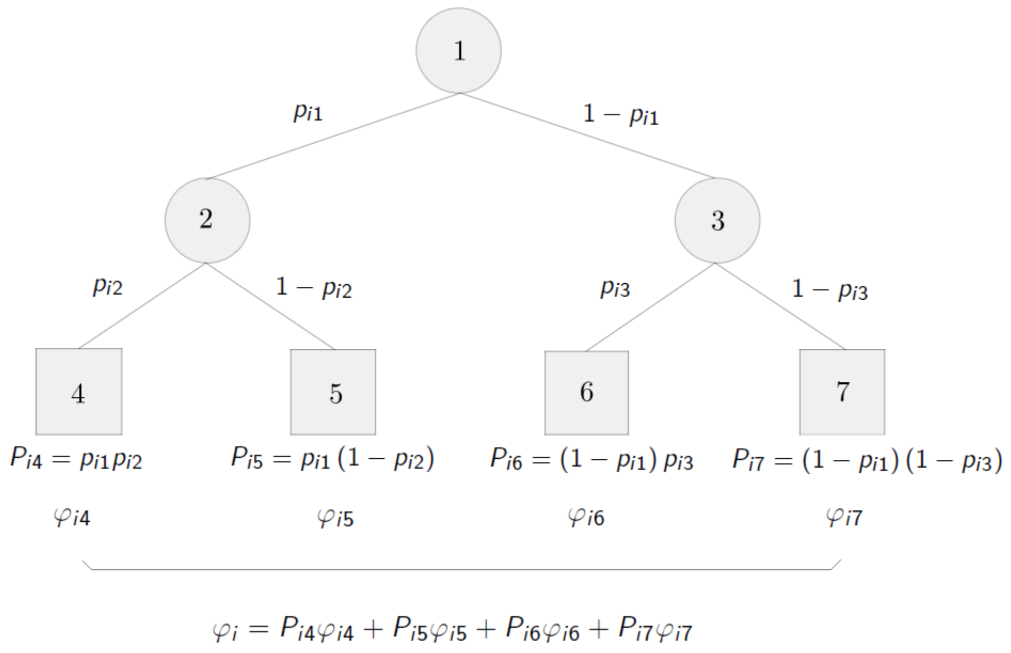
1
p_{i1}
$1 - p_{i1}$
2
3
p_{i2}
$1 - p_{i2}$
p_{i3}
$1 - p_{i3}$
4
5
6
7
$P_{i4} = p_{i1}p_{i2}$
$P_{i5} = p_{i1}(1 - p_{i2})$
$P_{i6} = (1 - p_{i1})\,p_{i3}$
$P_{i7} = (1 - p_{i1})(1 - p_{i3})$
φ_{i4}
φ_{i5}
φ_{i6}
φ_{i7}
$\varphi_i = P_{i4}\varphi_{i4} + P_{i5}\varphi_{i5} + P_{i6}\varphi_{i6} + P_{i7}\varphi_{i7}$

Figure 1: Sparse Optimal Randomized Regression Tree of depth $D = 2$.

to indicate by which branch an individual has to continue. Since binary trees are built, the Bernoulli distribution is appropriate, whose probability of success will be determined by the value of this CDF, evaluated over the vector of predictor variables. More precisely, at a given branch node t of the tree, an individual with predictor variables $\boldsymbol{x}_i$ will go either to the left or to the right child nodes with probabilities $F\left(\frac{1}{p}\boldsymbol{a}_{\cdot t}^T\boldsymbol{x}_i - \mu_t\right)$ and $1 - F\left(\frac{1}{p}\boldsymbol{a}_{\cdot t}^T\boldsymbol{x}_i - \mu_t\right)$, respectively, where $\boldsymbol{a}_{\cdot t}$ and μ_t are decision variables of the optimization problem that needs to be solved to build the S-ORRT. In Figure 1, p_{i1}, p_{i2}, p_{i3} and their complement to one denote such probabilities for the three branch nodes. With this, we have the probability of each individual in the sample falling into every leaf node. In Figure 1, P_{i4}, P_{i5}, P_{i6} and P_{i7} denote such probabilities. It remains to describe the prediction performed by an S-ORRT at each leaf node. S-ORRTs propose a generalization to the constant predictions that the classic approach for regression trees makes at leaf nodes. While in traditional approaches the subset of individuals falling into the same leaf node have associated the same constant prediction, in our approach we allow linear predictions instead. In Figure 1, φ_{i4}, φ_{i5}, φ_{i6} and φ_{i7} denote such linear predictions. As a result, the estimated outcome value for each individual in the tree would be the summation of these linear predictions, weighted by the probabilities of belonging to each leaf node under consideration. This is denoted by φ_i in Figure 1.

The following notation is required:

Parameters

D	depth of the binary tree,
p	number of predictor variables,
$\{(\boldsymbol{x}_i, y_i)\}_{i\in\mathcal{I}}$	training sample, where $\boldsymbol{x}_i \in [0,1]^p$ and $y_i \in \mathbb{R}$, with cardinality $\lvert\mathcal{I}\rvert$,
$F(\cdot)$	univariate continuously differentiable CDF, used to define the probabilities for an individual to go to the left or the right child node in the tree,
λ	sparsity regularization parameter.

Nodes

τ_B	set of branch nodes,
τ_L	set of leaf nodes,
$N_L(t)$	set of ancestor nodes of leaf node t whose left branch takes part in the path from the root node to leaf node t, $t \in \tau_L$,
$N_R(t)$	set of ancestor nodes of leaf node t whose right branch takes part in the path from the root node to leaf node t, $t \in \tau_L$.

Decision variables

$a_{jt} \in \mathbb{R}$	coefficient of predictor variable j in the oblique cut at branch node $t \in \tau_B$, or in the linear prediction at leaf node $t \in \tau_L$. The expressions $\boldsymbol{a}_B$ and $\boldsymbol{a}_L$ will denote the $p \times \lvert\tau_B\rvert$- and $p \times \lvert\tau_L\rvert$-matrices that involve these coefficients, respectively, $\boldsymbol{a}_B = (a_{jt})_{j=1,\ldots,p,\ t\in\tau_B}$ and $\boldsymbol{a}_L = (a_{jt})_{j=1,\ldots,p,\ t\in\tau_L}$. Let $\boldsymbol{a}$ denote the $p \times (\lvert\tau_B\rvert + \lvert\tau_L\rvert)$-matrix $\boldsymbol{a} = (a_{jt})_{j=1,\ldots,p,\ t\in\tau_B\cup\tau_L} = (\boldsymbol{a}_B, \boldsymbol{a}_L)$. Both notations will be used interchangeably when needed. The expressions $\boldsymbol{a}_{j\cdot}$ and $\boldsymbol{a}_{\cdot t}$ will denote the j-th row and the t-th column of $\boldsymbol{a}$, respectively,

$\mu_t \in \mathbb{R}$	location parameter at branch node $t \in \tau_B$, or intercept of the linear prediction at leaf node $t \in \tau_L$. The expressions $\boldsymbol{\mu}_B$ and $\boldsymbol{\mu}_L$ will denote the $\|\tau_B\|$- and $\|\tau_L\|$-vectors that involve these coefficients, respectively, $\boldsymbol{\mu}_B = (\mu_t)_{t\in\tau_B}$ and $\boldsymbol{\mu}_L = (\mu_t)_{t\in\tau_L}$. Let $\boldsymbol{\mu}$ denote the $(\|\tau_B\| + \|\tau_L\|)$-vector $\boldsymbol{\mu} = (\boldsymbol{\mu}_B, \boldsymbol{\mu}_L)$. Both notations will be used interchangeably when needed.

Probabilities

$p_{it}(\boldsymbol{a}_{\cdot t}, \mu_t)$	probability of individual i going down the left branch at branch node t. Its expression is $p_{it}(\boldsymbol{a}_{\cdot t}, \mu_t) = F\left(\frac{1}{p}\boldsymbol{a}_{\cdot t}^\top \boldsymbol{x}_i - \mu_t\right),\ i \in \mathcal{I},\ t \in \tau_B$,
$P_{it}(\boldsymbol{a}_B, \boldsymbol{\mu}_B)$	probability of individual i falling into leaf node t. Its expression is $P_{it}(\boldsymbol{a}_B, \boldsymbol{\mu}_B) = \prod_{t_l \in N_L(t)} p_{it_l}(\boldsymbol{a}_{\cdot t_l}, \mu_{t_l}) \prod_{t_r \in N_R(t)} (1 - p_{it_r}(\boldsymbol{a}_{\cdot t_r}, \mu_{t_r})), i \in \mathcal{I},\ t \in \tau_L$.

Predictions

$\varphi_{it}(\boldsymbol{a}_{\cdot t}, \mu_t)$	linear prediction of individual i at leaf node t. Its expression is $\varphi_{it}(\boldsymbol{a}_{\cdot t}, \mu_t) = \boldsymbol{a}_{\cdot t}^\top \boldsymbol{x}_i - \mu_t,\ i \in \mathcal{I},\ t \in \tau_L$,
$\varphi_i(\boldsymbol{a}, \boldsymbol{\mu})$	final prediction of individual i. Its expression is $\varphi_i(\boldsymbol{a}, \boldsymbol{\mu}) = \sum_{t\in\tau_L} P_{it}(\boldsymbol{a}_B, \boldsymbol{\mu}_B)\, \varphi_{it}(\boldsymbol{a}_{\cdot t}, \mu_t),\ i \in \mathcal{I}$. In other words, for an individual i, its prediction is a weighted average of the predictions φ_{it} along the different leaf nodes, where the weights in such average depend on the individual i.

With these parameters and decision variables, the S-ORRT reads as the following unconstrained NLCO problem:

$$\min_{\boldsymbol{a},\boldsymbol{\mu}} \left\{ \text{MSE}(\boldsymbol{a}, \boldsymbol{\mu}; \mathcal{I}) + \lambda \sum_{j=1}^{p} \|\boldsymbol{a}_{j\cdot}\|_\infty \right\}, \tag{1}$$

where

$$\mathrm{MSE}\left(\boldsymbol{a}, \boldsymbol{\mu}; \mathcal{I}\right) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \left(\varphi_i\left(\boldsymbol{a}, \boldsymbol{\mu}\right) - y_i\right)^2.$$

Each term in the objective function represents a criterion of S-ORRT. The first term, prediction accuracy, is equal to the mean squared error over the training sample between the actual response values and the predictions returned by S-ORRT. The second term controls sparsity. This is modeled by the inclusion of a penalization term that controls whether a given predictor variable is ever used across the whole tree. Recall that each predictor variable appears at both branch (in the oblique cuts) and leaf (in the linear predictions) nodes. Then, the ℓ_∞-norm is used as a group penalty function, by forcing all the coefficients linked to the same predictor variable to be shrunk simultaneously along all branch and leaf nodes.

Problem (1) is non-smooth due to the ℓ_∞-norm appearing in the objective function. Recall that F is assumed to be continuously differentiable, therefore MSE inherits smoothness. By rewriting the regularization term using new decision variables $\boldsymbol{\beta} = \left(\beta_j\right)_{j=1,\ldots,p}$, we can formulate S-ORRT as a smooth problem, thus solvable with standard continuous optimization solvers, as done in our computational section. Indeed, we have that Problem (1) is equivalent to

$$\min_{\boldsymbol{a}, \boldsymbol{\mu}, \boldsymbol{\beta}} \quad \left\{ \mathrm{MSE}(\boldsymbol{a}, \boldsymbol{\mu}; \mathcal{I}) \quad + \lambda \sum_{j=1}^{p} \beta_j \right\} \tag{2}$$

$$\text{s.t.} \quad -\beta_j \leq a_{jt} \leq \beta_j, \; j = 1, \ldots, p, \; t \in \tau_B \cup \tau_L. \tag{3}$$

Our approach can accommodate important desirable properties in regression tasks, such as performance constraints in critical groups. As a regression method, S-ORRT seeks a rule yielding a good overall prediction accuracy, although, at times, there are groups of individuals in which predicion errors are more critical. It is then more adequate not only to focus on the overall prediction accuracy, but, instead, obtaining an acceptable overall performance while ensuring a certain level of performance in those groups. S-ORRT is flexible enough to allow incorporating constraints on expected performance [9] over critical groups explicitly. Let $\mathcal{J}_1, \ldots, \mathcal{J}_r$ be different samples, possibly subsamples of $\mathcal{I}$. Given a threshold value $\rho_{\mathcal{J}_j}$ for the desired performance on sample $\mathcal{J}_j$, one can simply add the following constraints to

Problem (2)-(3):

$$\mathrm{MSE}(\boldsymbol{a}, \boldsymbol{\mu}; \mathcal{J}_j) \leq \rho_{\mathcal{J}_j}, \quad j = 1, \ldots, r.$$

3 Theoretical properties

In this section, some theoretical properties for S-ORRT are analyzed, with special focus on its most sparse case, when $\boldsymbol{a} = \boldsymbol{0}$ and thus none predictor variable is used in the predictions.

First, observe that, for any $\boldsymbol{a}$ and $\boldsymbol{\mu}_B$ fixed, Problem (1) can be easily reformulated as a linear regression problem. Indeed,

$$\varphi_i\left(\boldsymbol{a}, \boldsymbol{\mu}\right) = \sum_{t \in \tau_L} P_{it}\left(\boldsymbol{a}_B, \boldsymbol{\mu}_B\right)\left(\boldsymbol{a}_{\cdot t}^{\top} \boldsymbol{x}_i - \mu_t\right), \ i \in \mathcal{I},$$

and thus, defining

$$\eta_i\left(\boldsymbol{a}, \boldsymbol{\mu}_B\right) = \sum_{t \in \tau_L} P_{it}\left(\boldsymbol{a}_B, \boldsymbol{\mu}_B\right)\left(\boldsymbol{a}_{\cdot t}^{\top} \boldsymbol{x}_i - y_i\right), \ i \in \mathcal{I},$$

the MSE term in Problem (1) can be rewritten as

$$\frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \left(\eta_i\left(\boldsymbol{a}, \boldsymbol{\mu}_B\right) - \sum_{t \in \tau_L} P_{it}\left(\boldsymbol{a}_B, \boldsymbol{\mu}_B\right) \mu_t\right)^2,$$

or, in matrix form,

$$\frac{1}{|\mathcal{I}|} \left\| \boldsymbol{\eta}\left(\boldsymbol{a}, \boldsymbol{\mu}_B\right) - \boldsymbol{P}\left(\boldsymbol{a}_B, \boldsymbol{\mu}_B\right) \boldsymbol{\mu}_L \right\|^2,$$

where

$$\boldsymbol{\eta}\left(\boldsymbol{a}, \boldsymbol{\mu}_B\right) = \left(\eta_i\left(\boldsymbol{a}, \boldsymbol{\mu}_B\right)\right)_{i \in \mathcal{I}}$$

and

$$\boldsymbol{P}\left(\boldsymbol{a}_B, \boldsymbol{\mu}_B\right) = \left[\begin{array}{c} P_{it}\left(\boldsymbol{a}_B, \boldsymbol{\mu}_B\right) \end{array}\right]_{i \in \mathcal{I},\ t \in \tau_L}.$$

Then, minimizing MSE for $\boldsymbol{a}, \boldsymbol{\mu}_B$ fixed amounts to finding the Ordinary Least Squares solution with design matrix $\boldsymbol{P}\left(\boldsymbol{a}_B, \boldsymbol{\mu}_B\right)$ and response vector $\boldsymbol{\eta}\left(\boldsymbol{a}, \boldsymbol{\mu}_B\right)$. With this, the following is shown:

Proposition 1. *For $(\boldsymbol{a}^*, \boldsymbol{\mu}_B^*)$ fixed, $\boldsymbol{\mu}_L^*$ minimizes MSE$(\boldsymbol{a}^*, (\boldsymbol{\mu}_B^*, \boldsymbol{\mu}_L))$ if, and only if,*

$$\boldsymbol{P}^\top(\boldsymbol{a}_B^*, \boldsymbol{\mu}_B^*)\,\boldsymbol{\eta}(\boldsymbol{a}^*, \boldsymbol{\mu}_B^*) = \boldsymbol{P}^\top(\boldsymbol{a}_B^*, \boldsymbol{\mu}_B^*)\,\boldsymbol{P}(\boldsymbol{a}_B^*, \boldsymbol{\mu}_B^*)\,\boldsymbol{\mu}_L^*.$$

In particular, for the most sparse solution $\boldsymbol{a}^* = \boldsymbol{0}$, we have the following corollary.

Corollary 1. *For any $\boldsymbol{\mu}_B^*$, the vector $\boldsymbol{\mu}_L^* = \begin{pmatrix} -\bar{y}, & \cdots, & -\bar{y} \end{pmatrix}^\top$, with $\bar{y} = \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} y_i$, minimizes* $\mathrm{MSE}(\boldsymbol{0}, (\boldsymbol{\mu}_B^*, \boldsymbol{\mu}_L); \mathcal{I})$*, and then the prediction is $\varphi_i(\boldsymbol{0}, (\boldsymbol{\mu}_B^*, \boldsymbol{\mu}_L^*)) = \bar{y}$ for all $i \in \mathcal{I}$.*

Proof.

Observe that, by construction, $P_{it}(\boldsymbol{0}, \boldsymbol{\mu}_B^*)$ is independent of i. Hence, $\boldsymbol{P}(\boldsymbol{0}, \boldsymbol{\mu}_B^*)$ is a matrix with all its rows identical to a vector $u(\boldsymbol{\mu}_B^*)$, with $\sum_{t\in\tau_L} u_t(\boldsymbol{\mu}_B^*) = 1$.

Moreover, $\boldsymbol{\eta}(\boldsymbol{0}, \boldsymbol{\mu}_B^*) = -(y_i)_{i\in\mathcal{I}}$, and thus, for $\boldsymbol{a}^* = \boldsymbol{0}$ and $\boldsymbol{\mu}_B^*$, the vector $\boldsymbol{\mu}_L^* = \begin{pmatrix} -\bar{y}, & \cdots, & -\bar{y} \end{pmatrix}^\top$, satisfies the system of linear equations in Proposition 1.

$\square$

When λ is taken large enough in Problem (1), the first term related to the performance of the regression method becomes negligible and therefore $\boldsymbol{a}^*$ will shrink to $\boldsymbol{0}$. The tree with $\boldsymbol{a}^* = \boldsymbol{0}$ is the sparsest possible tree though possibly not yielding the best prediction accuracy, since none of the predictor variables is used to fit the model. It turns out that the solution $\boldsymbol{a}^* = \boldsymbol{0}$ is not only the limit case when λ tends to infinity, but it is actually obtained already for finite values of λ. This is shown in the following.

Proposition 2. *Let $\boldsymbol{a}^* = \boldsymbol{0}$, $\boldsymbol{\mu}_B^* \in \mathbb{R}^{|\tau_B|}$, and $\boldsymbol{\mu}_L^* = \begin{pmatrix} -\bar{y}, & \cdots, & -\bar{y} \end{pmatrix}^\top$. Let*

$$\lambda_0 = \max_{j=1,\ldots,p} \left\|\nabla_{\boldsymbol{a}_{j\cdot}} \mathrm{MSE}(\boldsymbol{0}, (\boldsymbol{\mu}_B^*, \boldsymbol{\mu}_L^*); \mathcal{I})\right\|_1.$$

Then, for any $\lambda \geq \lambda_0$, $(\boldsymbol{a}^*, (\boldsymbol{\mu}_B^*, \boldsymbol{\mu}_L^*))$ is a stationary point of Problem (1).

Proof.

First, let us consider the necessary optimality condiditions for $\boldsymbol{a}^*$. For $\lambda \geq \lambda_0$, we have, by construction, that

$$\lambda \geq \left\|\nabla_{\boldsymbol{a}_{j\cdot}} \mathrm{MSE}(\boldsymbol{0}, (\boldsymbol{\mu}_B^*, \boldsymbol{\mu}_L^*); \mathcal{I})\right\|_1, \ \forall j = 1, \ldots, p.$$

Hence,

$$-\nabla_{\boldsymbol{a}_{j\cdot}} \mathrm{MSE}\left(\boldsymbol{a}^*, \left(\boldsymbol{\mu}_B^*, \boldsymbol{\mu}_L^*\right); \mathcal{I}\right) \in \lambda \partial_{\boldsymbol{a}_{j\cdot}} \left(\|\boldsymbol{a}_{j\cdot}\|_\infty\right)\Big|_{\boldsymbol{a}_{j\cdot}=\mathbf{0}}$$

and thus,

$$-\nabla_{\boldsymbol{a}} \mathrm{MSE}\left(\boldsymbol{a}^*, \left(\boldsymbol{\mu}_B^*, \boldsymbol{\mu}_L^*\right); \mathcal{I}\right) \in \lambda \partial_{\boldsymbol{a}} \left(\sum_{j=1}^{p} \|\boldsymbol{a}_{j\cdot}\|_\infty\right)\Bigg|_{\boldsymbol{a}=\boldsymbol{a}^*},$$

i.e.:

$$\mathbf{0} \in \partial_{\boldsymbol{a}} \left(\mathrm{MSE}\left(\boldsymbol{a}^*, \left(\boldsymbol{\mu}_B^*, \boldsymbol{\mu}_L^*\right)\right) + \lambda \sum_{j=1}^{p} \|\boldsymbol{a}_{j\cdot}\|_\infty\right)\Bigg|_{\boldsymbol{a}=\boldsymbol{a}^*}.$$

For $\boldsymbol{a}^* = \mathbf{0}$, Corollary 1 shows that the chosen $\boldsymbol{\mu}_L^*$ minimizes MSE, and is thus optimal for Problem (1). In consequence, $\boldsymbol{\mu}_L^*$ satisfies the necessary optimality conditions.

Finally, let us analyze the optimality conditions for $\boldsymbol{\mu}_B = \boldsymbol{\mu}_B^*$. Observe that

$$\nabla_{\boldsymbol{\mu}_B} \mathrm{MSE}\left(\boldsymbol{a}, \left(\boldsymbol{\mu}_B, \boldsymbol{\mu}_L\right)\right) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} 2 \left(\varphi_i\left(\boldsymbol{a}, \boldsymbol{\mu}\right) - y_i\right) \nabla_{\boldsymbol{\mu}_B} \varphi_i\left(\boldsymbol{a}, \boldsymbol{\mu}\right).$$

Since $\boldsymbol{a}^* = \mathbf{0}$, $\nabla_{\boldsymbol{\mu}_B} \varphi_i\left(\boldsymbol{a}^*, \boldsymbol{\mu}^*\right)$ does not depend on $i \in \mathcal{I}$, say $\nabla_{\boldsymbol{\mu}_B} \varphi_i\left(\boldsymbol{a}^*, \boldsymbol{\mu}^*\right) = v$ for all $i \in \mathcal{I}$. Hence,

$$\nabla_{\boldsymbol{\mu}_B} \mathrm{MSE}\left(\boldsymbol{a}^*, \left(\boldsymbol{\mu}_B^*, \boldsymbol{\mu}_L^*\right)\right) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} 2 \left(\varphi_i\left(\boldsymbol{a}^*, \boldsymbol{\mu}^*\right) - y_i\right) v.$$

By Corollary 1, $\varphi_i\left(\boldsymbol{a}^*, \boldsymbol{\mu}^*\right) = \bar{y}$, and thus $\sum\limits_{i \in \mathcal{I}} \left(\varphi_i\left(\boldsymbol{a}^*, \boldsymbol{\mu}^*\right) - y_i\right) = 0$, implying

$$\nabla_{\boldsymbol{\mu}_B} \mathrm{MSE}\left(\boldsymbol{a}^*, \left(\boldsymbol{\mu}_B^*, \boldsymbol{\mu}_L^*\right)\right) = 0,$$

and the desired result follows.

□

4 Computational experiments

The aim of this section is to illustrate the performance of our sparse optimal randomized regression trees. The remainder of the section is structured as follows. Section 4.1 gives details on the procedure followed to test our approach. In Section 4.2 we discuss the prediction performance for S-ORRT, against several benchmarking regression methods. Finally, in

Section 4.3 we test S-ORRT for a range of values of the parameter λ, and illustrate that we are able to trade in some of our prediction accuracy for a gain in sparsity.

4.1 Setup

A collection of well-known real data sets from the UCI Machine Learning Repository [31] has been chosen for the computational experiments. Table 2 lists their names together with their number of observations and predictor variables.

Table 2: Information about the data sets considered.

Data set	$\vert\mathcal{I}\vert$	p
Housing	506	13
Red-wine	1599	11
White-wine	4898	11
Parkinson-motor	5874	16
Parkinson-total	5874	16
Ailerons	7153	40
Cpu-act	8192	21

Each data set has been randomly split into two subsets: the training subset (75%) and the test subset (25%). The corresponding tree model is built on the training subset and, then, two performance criteria, prediction accuracy and sparsity are measured. The prediction accuracy is evaluated by the out-of-sample R-squared (R^2) over the test subset:

$$R^2 = 1 - \frac{\mathrm{MSE}_{\mathrm{test}}}{\mathrm{V}_{\mathrm{test}}},$$

where $\mathrm{MSE}_{\mathrm{test}}$ is the mean squared error obtained by the regression method in question over the test subset and $\mathrm{V}_{\mathrm{test}}$ is the variance of the actual response vector of such subset. The R^2 is the proportion of the variance of the response variable that is explained by the model. The higher the R^2, the better the model in terms of prediction accuracy.

Sparsity is denoted by δ and reads as the percentage of predictor variables not used at any of the nodes, i.e., across the whole tree:

$$\delta = \frac{|\{\boldsymbol{a}_{j\cdot} = \mathbf{0},\ j = 1, \dots, p\}|}{p} \times 100.$$

The higher the δ, the better the model in terms of sparsity.

The training/testing procedure has been repeated ten times in order to avoid the effect of the initial split of the data. The results shown in Table 3 and Figure 2 represent the average of such ten runs for both performance criteria.

The logistic CDF has been chosen for our experiments:

$$F(\cdot) = \frac{1}{1 + \exp\left(-\left(\cdot\right)\gamma\right)},$$

with a large value of γ, namely, $\gamma = 512$. We will illustrate that this small level of randomization is enough for obtaining good results.

The S-ORRT smooth formulation (2)-(3) has been implemented using the `scipy.optimize` package [27] in Python 3.7 [37]. As solver, we have used the SLSQP method [29] that allows one to use gradient information. The predictor variables have been previously normalized to the $[0, 1]$ interval [32], and the decision variables $\boldsymbol{a}_B$ and $\boldsymbol{\mu}_B$ have been restricted to the $[-1, 1]$ interval. We have followed a multistart approach, where the process is repeated 500 times starting from different random initial solutions. Our experiments have been conducted on a PC, with an Intel$^{\circledR}$ Core$^{\text{TM}}$ i7-8550U CPU 1.80GHz processor and 8 GB RAM. The operating system is 64 bits.

4.2 Comparison of prediction performance

We first focus on illustrating prediction accuracy. S-ORRT at depths $D = 1,\ 2$ and 3 with $\lambda = 0$ is compared against three types of benchmark regression methods. The first type are standard regression methods, such as CART, the classic approach to build decision trees, with no restrictions on depth, and OLS. The second type is the benchmark regression method in sparsity, LASSO. Finally, in the third type we have two sophisticated tree-based regression methods benchmark in prediction accuracy, such as ORT-H LS in [16], a Mathematical Programming based version of CART that employs a local-search heuristic for building oblique trees with linear predictions at maximum depth $D = 10$; and Random Forest (RF), an ensemble of CARTs. Table 3 presents this comparison in terms of the average out-of-sample prediction accuracy R^2. The default parameter setting in `rpart` [40], `glmnet` [20] and `randomForest` [30] R packages have been used for running CART, OLS and LASSO, and RF, respectively. For ORT-H LS, the results are taken from [16], since open source implementations were not available.

Table 3: Comparison between S-ORRT with $\lambda = 0$, CART, OLS, LASSO, ORT-H LS and RF in terms of R-squared, R^2

Data set	Out-of-sample average R^2							
	S-ORRT $D = 1$	S-ORRT $D = 2$	S-ORRT $D = 3$	CART	OLS	LASSO	ORT-H LS	RF
Housing	0.7122	0.6523	0.7730	0.7416	0.7391	0.7401	0.8040	0.8759
Red-wine	0.3607	0.3721	0.3621	0.3055	0.3619	0.3605	0.3040	0.4874
White-wine	0.2960	0.3138	0.3171	0.2539	0.2714	0.2699	0.3490	0.5196
Parkinson-motor	0.1880	0.2213	0.2324	0.1020	0.0878	0.0900	0.2810	0.3426
Parkinson-total	0.1725	0.2209	0.2346	0.1294	0.0849	0.0863	0.3160	0.3545
Ailerons	0.8193	0.8168	0.8186	0.6466	0.8167	0.8173	0.8360	0.8211
Cpu-act	0.8171	0.9519	0.9330	0.9324	0.7272	0.7273	0.9840	0.9829

We start discussing the results for our S-ORRT with depth $D = 3$. S-ORRT generally outperforms CART. The improvements in the R^2 range from 2 to 17 percentage points (p.p.). This is the case for all data sets except for Cpu-act, where both methods are comparable. With respect to OLS and LASSO, both with comparable performance, S-ORRT obtains the best prediction accuracies for all the data sets considered, yielding increases in the R^2 up to 15 p.p. With respect to ORT-H LS, it generally outperforms S-ORRT. The improvements in the R^2 range from 1 to 8 p.p. This is the case for all data sets except for Red-wine, where S-ORRT manages to be superior to ORT-H LS in 6 p.p. RF reports the best average prediction accuracy for each of the data sets considered.

Regarding S-ORRT with depth $D = 1$ and 2, they outperform or are comparable to CART, OLS and LASSO in five and six data sets, respectively. This is not the case for Housing at both depths, as well as Cpu-act at depth $D = 1$. Similar conclusions can be drawn against RF and ORT-H LS. RF is the leader in terms of prediction accuracy, and ORT-H LS generally outperforms S-ORRT at depth $D = 1$ and 2, with the exception of Red-wine, where we are superior in 6 and 7 p.p., respectively.

In summary, these numerical results illustrate that, in terms of prediction accuracy, S-ORRT is better than the standard benchmark regression methods (CART and OLS) and the benchmark regression method in sparsity (LASSO). While sophisticated tree-based approaches (ORT-H LS and RF) show a slightly better prediction accuracy, our approach has a direct control on the sparsity across the whole tree, as illustrated in the next section.

4.3 Prediction accuracy and sparsity tradeoff

The aim of this section is to illustrate that, in contrast to sophisticated tree-based regression methods that employ greedy or local-search approaches, such as RF and ORT-H LS, our S-ORRT is able to trade in some of its prediction accuracy for a gain in sparsity. We have tested our S-ORRT in the Ailerons data set. We have run the S-ORRT at depth $D = 2$ for the grid of values $\lambda = \left\{ \frac{2^r}{p},\ -12 \leq r \leq 12,\ r \in \mathbb{Z} \right\}$. We start solving the optimization problem with $\lambda = \frac{2^{-12}}{40}$, where the multistart approach uses 500 random initial solutions. We continue solving the optimization problem for the next value of λ, and we feed the corresponding optimization problem with the 500 solutions resulting from the problem solved for the previous value.

Figure 2 depicts these results, by showing simultaneously the sparsity measure δ (blue solid line) and the prediction accuracy measure R^2 (red dashed line) as a function of the grid of the λ's considered. As expected, the larger the λ, the larger the δ. The sparsest tree is obtained for large values of λ, where the best solution in terms of sparsity is obtained but the worst possible one in terms of prediction accuracy. Clearly, the R^2 stays almost constant for $\lambda \leq \frac{2^7}{40}$, while the sparsity improves from 0% to 60%. For larger values of λ, our S-ORRT keeps improving sparsity at the cost of R^2. These results illustrate that S-ORRT successfully makes the tradeoff between both performance criteria.

5 Conclusions and future research

In recent years, several papers have focused on building decision trees in which the greedy suboptimal construction approach is replaced by solving an optimization problem, usually in integer variables. In this paper, we have proposed a new continuous optimization-based approach to build regression trees. Unlike CART, we can directly model desirable properties such as sparsity. The computational experience reported shows that our method outperforms

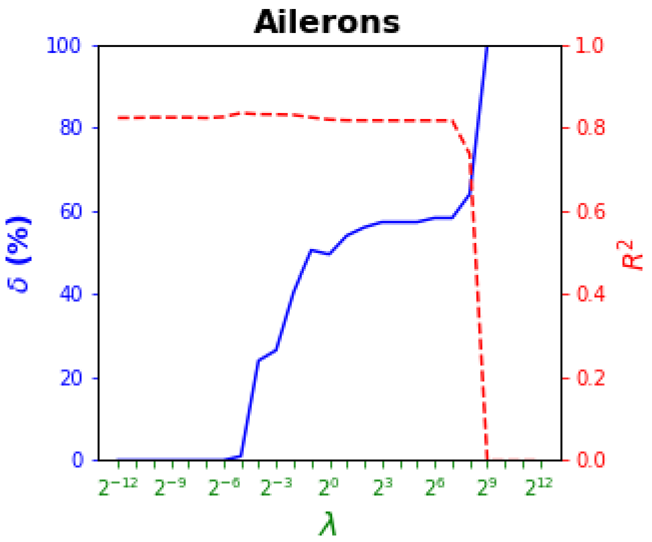

Figure 2: Graphical representation, for Ailerons data set, of the average percentage of predictor variables per tree, δ, together with the average R-squared obtained, R^2, as a function of the values of λ considered in the S-ORRT construction.

CART, as well as OLS and LASSO, in terms of prediction accuracy. Moreover, we show to be able to easily trade in some of our prediction accuracy for a gain in sparsity, unlike more sophisticated tree-based approaches such as RF.

Several extensions to our approach are attractive. First, the linear prediction made at each leaf node can easily be extended to a non-linear one. This would be obtained by simply replacing the linear functions φ_{it} with other functions, such as those in a Generalized Additive Model. Second, it is known that standard Regression Analysis seeks an estimate of the conditional mean of the response variable, given the predictor vector, which is found by minimizing the mean squared error, as proposed in this paper. Nevertheless, it would be interesting to infer other characteristics of the distribution of the response variable, such as the conditional quantiles. An appropriate setting of our approach that considers quantile regression [35] requires a nontrivial design. Third, a bagging scheme of our approach, where the collection of trees is solved simultaneously in order to have a global control on sparsity, is also an interesting open question. A parallelization framework would be suitable to make the training of a collection of trees tractable.

Acknowledgements

This research has been financed in part by research projects EC H2020 MSCA RISE NeEDS (Grant agreement ID: 822214), FQM-329 and P18-FR-2369 (Junta de Andalucía, Spain). This support is gratefully acknowledged.

References

[1] S. Athey. The impact of machine learning on economics. In *The Economics of Artificial Intelligence: An Agenda*. University of Chicago Press, 2018.

[2] B. Baesens, R. Setiono, C. Mues, and J. Vanthienen. Using neural network rule extraction and decision tables for credit-risk evaluation. *Management Science*, 49(3):312–329, 2003.

[3] K. P. Bennett and J. Blue. Optimal decision trees. *Rensselaer Polytechnic Institute Math Report*, 214, 1996.

[4] D. Bertsimas, J. Dunn, and A. Paschalidis. Regression and classification using optimal decision trees. In *Undergraduate Research Technology Conference (URTC), 2017 IEEE MIT*, pages 1–4, 2017.

[5] M. Better, F. Glover, and M. Samorani. Classification by vertical and cutting multi-hyperplane decision tree induction. *Decision Support Systems*, 48(3):430–436, 2010.

[6] G. Biau and E. Scornet. A random forest guided tour. *TEST*, 25(2):197–227, 2016.

[7] R. Blanquero, E. Carrizosa, C. Molero-Río, and D. Romero Morales. Optimal Randomized Classification Trees. `https://www.researchgate.net/publication/326901224_Optimal_Randomized_Classification_Trees`, 2018.

[8] R. Blanquero, E. Carrizosa, C. Molero-Río, and D. Romero Morales. Sparsity in optimal randomized classification trees. *European Journal of Operational Research*, 284(1):255 – 272, 2020.

[9] R. Blanquero, E. Carrizosa, P. Ramírez-Cobo, and M. R. Sillero-Denamiel. A cost-sensitive constrained lasso. Forthcoming in *Advances in Data Analysis and Classification*, 2020. https://doi.org/10.1007/s11634-020-00389-5.

[10] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[11] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees.* CRC press, 1984.

[12] E. Carrizosa, B. Martín-Barragán, and D. Romero Morales. Detecting relevant variables and interactions in supervised classification. *European Journal of Operational Research*, 213(1):260–269, 2011.

[13] I. Chikalov, S. Hussain, and M. Moshkov. Bi-criteria optimization of decision trees with applications to data analysis. *European Journal of Operational Research*, 266(2):689–701, 2018.

[14] H. Deng and G. Runger. Feature selection via regularized trees. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2012.

[15] H. Deng and G. Runger. Gene selection with guided regularized random forest. *Pattern Recognition*, 46(12):3483–3489, 2013.

[16] J. Dunn. *Optimal trees for prediction and prescription.* PhD thesis, Massachusetts Institute of Technology, 2018.

[17] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15(1):3133–3181, 2014.

[18] M. Firat, G. Crognier, A. Gabor, C. Hurkens, and Y. Zhang. Column generation based math-heuristic for classification trees. *Computers & Operations Research*, pages 1810–06684, 2019.

[19] A. Freitas. Comprehensible classification models: a position paper. *ACM SIGKDD Explorations Newsletter*, 15(1):1–10, 2014.

[20] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[21] R. Genuer, J.-M. Poggi, C. Tuleau-Malot, and N. Villa-Vialaneix. Random Forests for Big Data. *Big Data Research*, 9:28–46, 2017.

[22] B. Goodman and S. Flaxman. European Union regulations on algorithmic decision-making and a "right to explanation". *arXiv preprint arXiv:1606.08813*, 2016.

[23] O. Günlük, J. Kalagnanam, M. Menickelly, and K. Scheinberg. Optimal Decision Trees for Categorical Data via Integer Programming. *arXiv preprint arXiv:1612.03225v2*, 2018.

[24] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. New York: Springer, second edition, 2009.

[25] X. Hu, C. Rudin, and M. Seltzer. Optimal sparse decision trees. *arXiv preprint arXiv:1904.12847*, 2019.

[26] L. Hyafil and R. L. Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1):15–17, 1976.

[27] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001.

[28] J. Jung, C. Concannon, R. Shroff, S. Goel, and D. G. Goldstein. Simple rules for complex decisions. *arXiv preprint arXiv:1702.04690*, 2017.

[29] D. Kraft. A software package for sequential quadratic programming. Technical report, DFVLR-FB 88-28, DLR German Aerospace Center — Institute for Flight Mechanics, Köln, Germany, 1988.

[30] A. Liaw and M. Wiener. Classification and Regression by randomForest. *R News*, 2(3):18–22, 2002.

[31] M. Lichman. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml. University of California, Irvine, School of Information and Computer Sciences, 2013.

[32] D.-C. Luor. A comparative assessment of data standardization on support vector machine for classification problems. *Intelligent Data Analysis*, 19(3):529–546, 2015.

[33] D. Martens, B. Baesens, T. Van Gestel, and J. Vanthienen. Comprehensible credit scoring models using rule extraction from support vector machines. *European Journal of Operational Research*, 183(3):1466–1476, 2007.

[34] B. Martín-Barragán, R. Lillo, and J. Romo. Interpretable support vector machines for functional data. *European Journal of Operational Research*, 232(1):146–155, 2014.

[35] N. Meinshausen. Quantile regression forests. *Journal of Machine Learning Research*, 7:983–999, 2006.

[36] N. Narodytska, A. Ignatiev, F. Pereira, J. Marques-Silva, and I. RAS. Learning optimal decision trees with SAT. In *IJCAI*, pages 1362–1368, 2018.

[37] Python Core Team. Python: A dynamic, open source programming language. Python Software Foundation, 2015.

[38] G. Ridgeway. The pitfalls of prediction. *National Institute of Justice Journal*, 271:34–40, 2013.

[39] S. Ruggieri. Complete search for feature selection in decision trees. *Journal of Machine Learning Research*, 20(104):1–34, 2019.

[40] T. Therneau, B. Atkinson, and B. Ripley. *rpart: Recursive Partitioning and Regression Trees*, 2015.

[41] R. Tibshirani, M. Wainwright, and T. Hastie. *Statistical Learning with Sparsity. The Lasso and Generalizations.* Chapman and Hall/CRC, 2015.

[42] B. Ustun and C. Rudin. Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, 102(3):349–391, 2016.

[43] S. Verwer and Y. Zhang. Learning decision trees with flexible constraints and objectives using integer optimization. In *International Conference on AI and OR Techniques*

in Constraint Programming for Combinatorial Optimization Problems, pages 94–103. Springer, 2017.

[44] S. Verwer and Y. Zhang. Learning optimal classification trees using a binary linear program formulation. In *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, volume 33, pages 1625–1632. AAAI Press, 2019.

[45] L. Yang, S. Liu, S. Tsoka, and L. G. Papageorgiou. A regression tree approach using mathematical programming. *Expert Systems with Applications*, 78:347–357, 2017.