Least-Squares-Based Construction Algorithm for Oblique and Mixed Regression Trees

Marvin Schöne, Martin Kohlhase

Center for Applied Data Science Gütersloh, FH Bielefeld
Interaktion 1, 33619 Bielefeld
E-Mail: {marvin.schoene, martin.kohlhase}@fh-bielefeld.de

1 Introduction

As part of Smart Factories, industrial processes must be optimized in terms of efficiency, flexibility and process reliability. This is primarily achieved by *Advanced Analytics*, where data-driven models are used to analyze, describe and predict process behavior [1]. In this way, new process knowledge is gained and used, for instance, to adjust the operation mode of the process or reduce defects and quality problems [2]. These models need to fulfill certain requirements to be applicable in an industrial environment. In order to ensure a reliable operation of the plant and to enable optimization, they must be highly accurate. Furthermore, to gain process knowledge and confidence towards the operators and to fix model uncertainties more easily, they must be interpretable.

Decision Trees are a model class that can fulfill these requirements [3]. They are algorithmic constructed and represented as a top-down directional acyclic graph, consisting of decision nodes and terminal leaves. This graph is specified as a tree, which starts with a single decision node, the root, and ends up in multiple terminal leaves. To predict an output variable $\hat{y}$ (e.g. process behavior), an unlabeled sample $\mathbf{x} = [x_1 \ \dots \ x_M]$, which consists of M input variables x_m with $m \in \mathbb{N} \mid 1 \leq m \leq M$, must pass through the tree until a terminal leaf is reached. Each decision node contains a test function, that is applied at $\mathbf{x}$ and effects the path that $\mathbf{x}$ passes through the tree. The test functions are usually formulated as univariate splitting criteria $x_m \leq c$ for $c \in \mathbb{R}$

DOI: 10.58895/ksp/1000124139-13 erschienen in:
Proceedings – 30. Workshop Computational Intelligence: Berlin, 26. - 27. November 2020
DOI: 10.58895/ksp/1000124139 | https://www.ksp.kit.edu/site/books/m/10.58895/ksp/1000124139/

or $x_m \in \mathscr{B}$ for $\mathscr{B} \subset \mathscr{A}$ with a numerical threshold value c or a subset $\mathscr{B}$ of a merge of categorical attributes $\mathscr{A}$. Each terminal leaf contains a local model for prediction, that is only valid in a certain partition of the input space defined by trees' structure. Because of the rule-based structure, trees are human readable and easy to interpret. They can predict both numerical (*Regression Tree*) and categorical (*Classification Tree*) output variables. In addition, numerical and categorical input variables as well as missing input values can be handled and the importance of input variables can be measured [3, 4, 5].

However, especially for *Regression Trees* with univariate splitting criteria, there are limitations which can result in lower model accuracy and interpretability [4, 6]. Univariate splitting criteria depend on a single input variable, resulting in axis-orthogonal splits that limit model flexibility. Depending on the process function, this leads to lower model accuracy and, if simple local models are used, to a larger tree, which reduces interpretability [7]. To overcome these issues, multivariate splitting criteria $\sum_{m=1}^{M} \beta_m x_m \leq c$ with M coefficients β_m can be used to construct axis-oblique splits. The resulting tree is called *Oblique Regression Tree* or, if uni- and multivariate splitting criteria are used, *Mixed Regression Tree* [8]. The direction of an axis-oblique split has to adapt to the curvature of the function and is given by its coefficients β_m [8, 9, 10]. Furthermore, to maintain interpretability, to avoid overfitting and to overcome the curse of dimensionality, an efficient and generalized approach is necessary.

In this paper, a novel algorithm to construct *Mixed* and *Oblique Regression Trees* is presented. To determine an axis-oblique split direction adapted to the curvature in a partition, a first-order *Least Squares Regression* (LSR) model is used. This model is limited to significant input variables to describe this curvature, which maintains interpretability and generalization. The input variables are selected by analyzing the residuals of the resulting splitting model, which additionally weakens the curse of dimensionality. To construct the local models for prediction, stepwise regression is used. In Section 2, common algorithms for the construction of *Regression Trees* are explained. The proposed algorithm is presented in Section 3 and tested in Section 4 in an extensive experimental analysis with both synthetic and real-world data. Moreover, the results are compared with a state-of-the-art construction algorithm. At the end, Section 5 summarizes the paper and gives an overview on further research.

2 Construction Algorithms for Regression Trees

In this Section, the functionality of algorithms to construct *Regression Trees* is explained. The functionality is described in more detail for the common algorithms SUPPORT [11], CART [12], GUIDE [6] and PHDRT [10], which generate the eponymous trees.

Regression Trees are constructed by a *divide-and-conquer* strategy, which splits a set of N labeled samples $\mathscr{D} = \{\mathbf{X}, \mathbf{y}\}$ with $\mathbf{X} = [\mathbf{x}_1 \ \dots \ \mathbf{x}_N]^{\mathrm{T}}$ and the corresponding labels $\mathbf{y} = [y_1 \ \dots \ y_N]^{\mathrm{T}}$ recursively into smaller subsets $\mathscr{D}_k$ until a stopping criterion is reached. Each set of labeled samples $\mathscr{D}_k$ is represented as a node t_k with $k \in \{1, 2, \dots, K\}$ within the tree T, that consists of $|T| = K$ nodes [5].

The recursive splitting process to construct a tree is shown in Figure 1 and starts with the entire data set $\mathscr{D}_1 = \mathscr{D}$, represented by the root t_1. At first, in step a) a stopping rule for the node t_k is check. The stopping rule ensures that only meaningful splits of $\mathscr{D}_k$ are performed and the size of the tree is limited. A common stopping rule is a lower bound of the number of samples in a node, which is used in all four algorithms [5].

In step b) of Figure 1, the node t_k becomes a terminal leaf $\tilde{t}_k$ when splitting is stopped. Each terminal leaf represents a certain partition of the input space and contains a local model $\hat{y}_{\tilde{t}_k}(\mathbf{x}) \in \mathbb{R}$, that approximates the function within that partition. The local models of GUIDE and PHDRT are first-order multiple regression models and those of SUPPORT are third-order polynomial regression models [11, 6, 10]. Furthermore, SUPPORT combines all local models by a weighted average to create a continuous model output. For the local models of SUPPORT and PHDRT, all input variables are used. In contrast, GUIDE limits the local models to significant input variables using stepwise regression. The local models of CART are constant values, which are determined by the mean value of $\mathbf{y}$ in that partition [12].

The node t_k will be further split if the stopping rule is not fulfilled. For this purpose, in step c) of Figure 1 the input variable(s) x_m and the threshold value c or subset $\mathscr{B}$ to construct an uni- or multivariate splitting criterion are selected. The components x_m and c or $\mathscr{B}$ are selected in a way that the impurity is

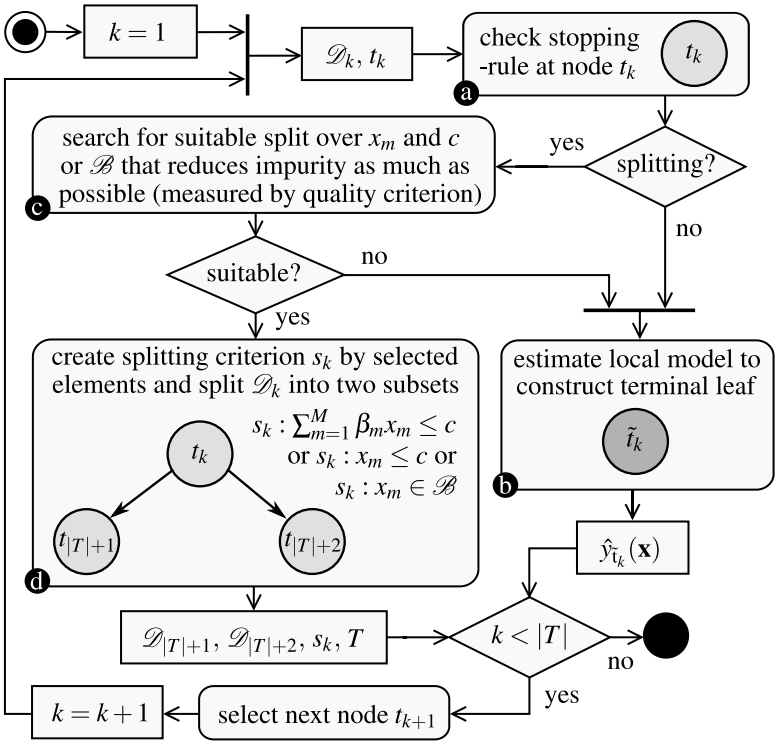


Figure 1: Recursive splitting process of a construction algorithm to create an unpruned tree T.

reduced as much as possible by the resulting splitting criterion [5, 13]. In *Regression Trees* the impurity is described e.g. by the degree of non-linearity in a partition, which is measured by a specific quality criterion. To measure the non-linearity in a partition, the error of a linear approximation can be used. To split a node, CART uses both uni- and multivariate criteria, which are either selected by a brute force method (univariate) or by a heuristic-based selection method (multivariate) called *Linear Combination Search Algorithm* [12]. In contrast, SUPPORT is limited to univariate splitting criteria and numerical input variables. For split selection, the samples within a node are approximated by multiple linear regression and divided into two sample groups with positive and negative signs of residuals. Due to statistical tests for differences in mean and variance between these two sample groups, dependencies between input

variables and residuals are analyzed to select the most significant x_m. The threshold value c is determined by averaging the means of the two groups of x_m [11]. The splitting method of GUIDE is similar to SUPPORTs' and differs in the use of a χ^2-independence tests, an interaction test between input variables and the ability to handle categorical input variables. Furthermore, due to a bootstrap-based bias correction, the significance of input variables is more comparable. Threshold values c are either determined by the median or mean of x_m and subsets $\mathscr{B}$ for a categorical input variable by a heuristic [6]. The splitting of PHDRT is limited to numerical input variables and multivariate criteria, which are determined by the first component of *principal Hessian Directions* (PHD). This component describes the direction in which the function to be approximated has the greatest curvature. To select a threshold value, the residuals of a multiple linear regression model are split into two partitions and approximated by one linear regression model each, using the first component of PHD as an input variable. The balance of the partitions is adjusted so that both linear models approximate the residuals with a similar standard deviation. Finally, the point of intersection is taken as the threshold value [10].

If a suitable split is determined, in d) of Figure 1 the splitting criterion s_k is constructed based on the selected components. In addition, the node is split into two nodes $t_{|T|+1}$ and $t_{|T|+2}$ containing the subsets $\mathscr{D}_{|T|+1}$ and $\mathscr{D}_{|T|+2}$. Recursive splitting is completed when no more nodes can be split [5, 13].

Further approaches to limit the size of the tree are pre- and postpruning techniques. Prepruning is closely related to the stopping rule and limits the size during the construction. In contrast, postpruning is applied after the construction and prunes an oversized tree backwards to a more generalized one. For this purpose, PHDRT stops splitting if the first component of PHD is insignificant, which is more similar to a stopping rule than to a prepruning technique [10]. The complexity of SUPPORT is limited by a prepruning technique, which uses cross validation to check whether a subtree can be created from t_k that significantly improves model quality [11]. Both CART and GUIDE limit the size by a postpruning technique called *Minimal Cost Complexity Pruning*, which uses cross validation to evaluate the generalization capabilities of different subtrees during the pruning [12, 6]. In the following, the proposed least-squares-based tree construction algorithm is explained in more detailed.

3 Least-Squares-Based Construction Algorithm

Model quality of *Regression Trees* can be improved by an extension to *Oblique* or *Mixed Regression Trees* using multivariate splitting criteria. To obtain the advantages of *Regression Trees*, a trade-off must be found between an increase in model accuracy and a loss of interpretability. This is a challenging task. In order to achieve this, the axis-oblique split direction of a multivariate splitting criterion must adapt to the function gradient $\nabla f(\mathbf{x})$ in a curvature area. Furthermore, the complexity of this criterion must be limited in such a way that interpretability is maintained. To determine this criterion with an appropriate computational effort, the curse of dimensionality must be weakened [6, 4, 8].

To construct the splitting criterion, the proposed algorithm uses the coefficients of a first-order LSR model $\hat{y}_{s_k}(\mathbf{x})$. The input variables of $\hat{y}_{s_k}(\mathbf{x})$ are selected in a way that $\hat{y}_{s_k}(\mathbf{x})$ adapts to the gradient $\nabla f(\mathbf{x})$ in the area of curvature in that partition. This is performed by a forward selection method (FSM) and depending on the number of selected input variables either an uni- or multivariate splitting criterion is constructed. A model with a single input variable constructs an univariate splitting criterion and a model with multiple input variables a multivariate splitting criterion. The split direction is defined by the contour lines, contour planes or contour hyperplanes (depending on M) of $\hat{y}_{s_k}(\mathbf{x})$ and by selecting a suitable output value of $\hat{y}_{s_k}(\mathbf{x})$ as a threshold, the position of the split is adjusted, which is explained in Subsection 3.1. The quality of the resulting split is measured by a criterion which analyzes the residuals of $\hat{y}_{s_k}(\mathbf{x})$ using a hinge function $h(\hat{y}_{s_k})$. Due to a reduction of the search space to the one-dimensional space of residuals, the FSM overcomes the curse of dimensionality. Furthermore, by limiting the number of significant input variables to a maximum of λ, interpretability and generalization is maintained. In Subsection 3.2 the FSM is presented in more detail.

If the sample size is too small or an insufficient improvement in model quality is achieved, splitting is stopped and a local model for prediction is determined. The local models are also determined by LSR and a FSM, which is explained in Subsection 3.3. To control the size of tree called *Least Squares Regression Tree* (LSRT), the technique *Minimal Cost Complexity Pruning* is used [12]. Finally, Subsection 3.4 shows the structure of LSRT using a practical example.

3.1 Axis-Oblique Split Direction

To get an axis-oblique split direction that is adapted to the gradient $\nabla f(\mathbf{x})$ in the area of curvature within a partition, a direction orthogonal to $\nabla f(\mathbf{x})$ has to be determined. This is achieved using a first-order LSR model

$$\hat{y}_{s_k}(\mathbf{x}) = \beta_0 + \sum_{m=1}^{M} \beta_m x_m \ . \tag{1}$$

If the non-linearity in a local area is not excessive, a direction orthogonal to $\nabla f(\mathbf{x})$ is obtained in this way. The coefficients of the model are determined by

$$\boldsymbol{\beta} = [\beta_0 \ \ldots \ \beta_M]^{\mathrm{T}} = (\mathrm{X}^{\mathrm{T}}\mathrm{X})^{-1}\mathrm{X}^{\mathrm{T}}\mathbf{y} \tag{2}$$

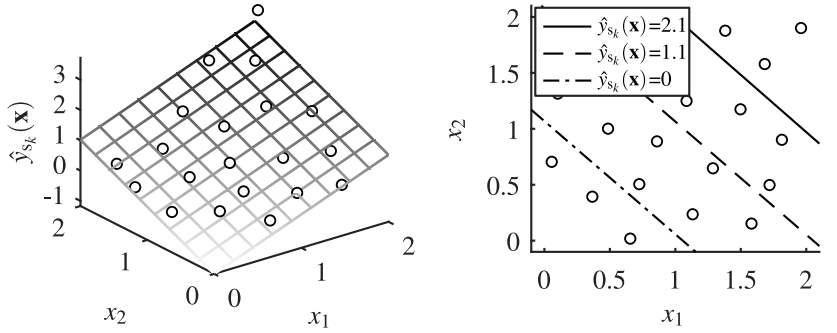
using the expanded $N \times (1+M)$ predictor matrix

$$\mathrm{X} = \begin{pmatrix} 1 & \mathbf{x}_1 \\ 1 & \mathbf{x}_2 \\ \vdots & \vdots \\ 1 & \mathbf{x}_N \end{pmatrix} = \begin{pmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,M} \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,M} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & x_{N,2} & \cdots & x_{N,M} \end{pmatrix}, \tag{3}$$

that consists of N samples $\mathbf{x}_n$ and an additional column of ones to determine the constant part β_0 of the LSR model [9].

Figure 2a shows a first-order LSR model $\hat{y}_{s_k}(\mathbf{x})$, that was trained on the 20 samples generated by a test function $f(\mathbf{x}) = x_1 x_2$. A contour line for the constant model output $\hat{y}_{s_k}(\mathbf{x}) = \alpha$ is formed by the various input combinations which result in α and runs as an axis-oblique border through the input space. The direction of the contour line results from the coefficients $[\beta_1 \ \ldots \ \beta_M]^{\mathrm{T}}$ and is orthogonal to $\nabla\hat{y}_{s_k}(\mathbf{x})$, which is presented in Figure 2b. This Figure shows three possible contour lines resulting from $\alpha \in \{0, 1.1, 2.1\}$ and $\hat{y}_{s_k}(\mathbf{x})$ in Figure 2a. The contour lines are splitting the input space into two partitions and by varying α, they are parallel shifted. This allows to determine a suitable threshold value for splitting. Finally, to construct the multivariate splitting criterion

$$\sum_{m=1}^{M} \beta_m x_m \leq \alpha - \beta_0 \ , \tag{4}$$



(a) Model output $\hat{y}_{s_k}(\mathbf{x})$ of a first-order LSR model (grid), trained on 20 samples generated from $f(\mathbf{x}) = x_1 x_2$.

(b) Three oblique splits that result from the contour lines of the model and are orthogonal to $\nabla\hat{y}_{s_k}(\mathbf{x})$.

Figure 2: Construction of axis-oblique splits using contour lines of a LSR model.

the constant part β_0 of the LSR model is subtracted. The threshold value $c = \alpha - \beta_0$ as well as the input variables to construct the LSR model are determined by a FSM, which is explained in the following Subsection.

3.2 Split Selection

In order to construct an uni- or multivariate splitting criterion with regard to the requirements of approximation capability, interpretability and generalization, suitable input variables for the LSR model and a suitable threshold value c must be selected. This is achieved by the FSM presented in Figure 3.

At first, the local optimal quality value $\gamma^* \in \mathbb{R}$, a maximum number of input variables $\lambda \in \mathbb{N} \mid 1 \leq \lambda \leq M$ to limit the complexity of the splitting criterion and the index m are initialized. Furthermore, the selected input variables $\mathbf{x}^* \in \mathbb{R}^i$ to construct the final splitting criterion are initialized with $\mathbf{x}^* = [1]$ for the constant part β_0. Each forward iteration performs a greedy search over all unselected input variables to extend an existing splitting criterion (resulting from $\mathbf{x}^*$) by a local optimal candidate input variable x_m. To identify the local optimal candidate during the greedy search, the quality of the splitting crite-

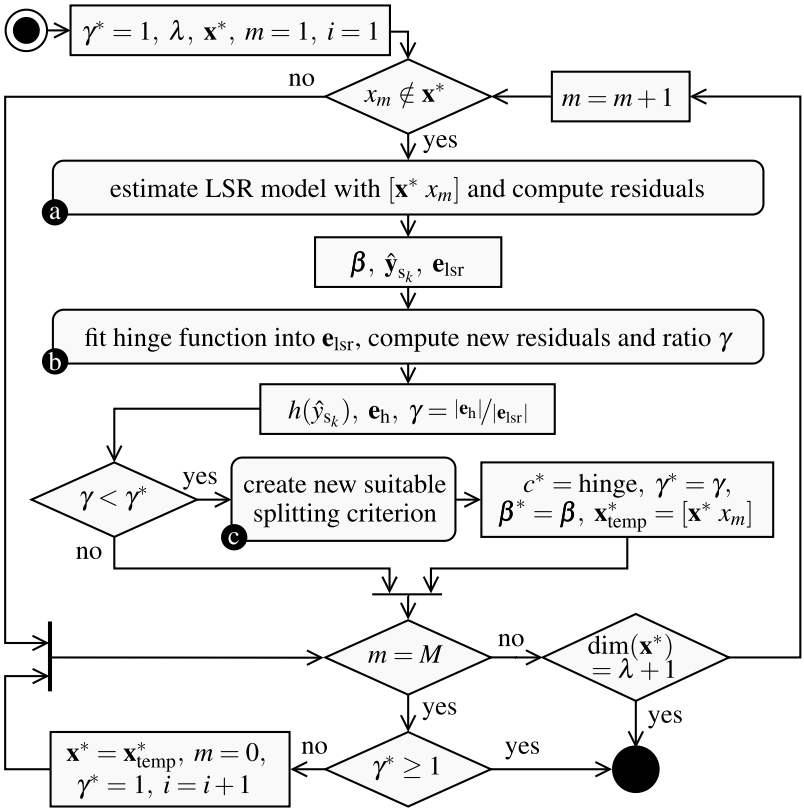


Figure 3: Activity diagram of the FSM to construct a suitable uni- or multivariate splitting criterion.

rion resulting from an extension with x_m is measured using a specific quality criterion.

To measure the quality which results from the extension, in step a) a LSR model $\hat{y}_{s_k}(\mathbf{x}^*, x_m)$ for splitting is determined. This model is constructed based on the previously selected input variables $\mathbf{x}^*$ and the candidate x_m. The residuals [9]

$$\mathbf{e}_{\text{lsr}} = [e_1\ \ldots\ e_N]^{\text{T}} = \mathbf{y} - \hat{\mathbf{y}}_{s_k} = [y_1 - \hat{y}_{s_k,1}\ \ldots\ y_N - \hat{y}_{s_k,N}]^{\text{T}} \tag{5}$$

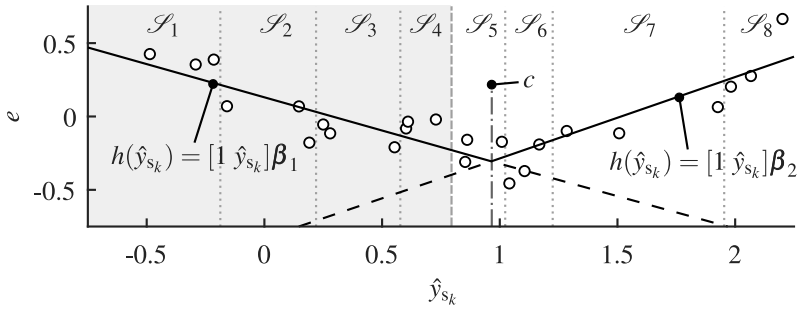


Figure 4: Example of the process to measure the quality of a splitting criterion. The residuals e of a LSR model $\hat{y}_{s_k}(\mathbf{x}^*)$ are approximated by a hinge function $h(\hat{y}_{s_k})$ (solid line), which consists of two LSR models trained by different subsets $\mathscr{S}_k$. The quality is measured by the improvements in approximation capability by $h(\hat{y}_{s_k})$.

of the model output $\hat{y}_{s_k,n} = \hat{y}_{s_k}(\mathbf{x}_n^*, x_{n,m})$ with $n \in \{1,\dots N\}$ are computed and in step b) approximated by a hinge function

$$h(\hat{y}_{s_k}) = \min([1\ \hat{y}_{s_k}]\boldsymbol{\beta}_1, [1\ \hat{y}_{s_k}]\boldsymbol{\beta}_2) \quad \text{or} \quad h(\hat{y}_{s_k}) = \max([1\ \hat{y}_{s_k}]\boldsymbol{\beta}_1, [1\ \hat{y}_{s_k}]\boldsymbol{\beta}_2)\ . \quad (6)$$

The hinge function consists of two local linear LSR models $\boldsymbol{\beta}_i = [\beta_0\ \beta_1]^{\mathrm{T}}\ \forall\ i \in \{1,2\}$, which are joined together by a hinge point [9, 14].

Figure 4 shows a hinge function (solid line), that was determined by $N = 24$ samples $\mathscr{E} = \{\hat{\mathbf{y}}_{s_k}, \mathbf{e}_{\mathrm{lsr}}\}$. To construct $h(\hat{y}_{s_k})$, the samples are ordered and segmented into K subsets $\mathscr{S}_k$, containing an equal number of samples. This segmentation helps to overcome the challenging effects of skewed data. The subsets resulting from the segmentation are grouped into $\mathscr{E}_{\mathrm{left}}$ and $\mathscr{E}_{\mathrm{right}}$ to determine $\boldsymbol{\beta}_1$ with $\mathscr{E}_{\mathrm{left}}$ and $\boldsymbol{\beta}_2$ with $\mathscr{E}_{\mathrm{right}}$. This is done iterative by changing the proportion of the groups until a stopping criterion is reached. In Figure 4, $K = 8$ subsets are used. First, the models are determined by two balanced groups $\mathscr{E}_{\mathrm{left}} \supset \{\mathscr{S}_1, \mathscr{S}_2, \mathscr{S}_3, \mathscr{S}_4\}$ and $\mathscr{E}_{\mathrm{right}} \supset \{\mathscr{S}_5, \mathscr{S}_6, \mathscr{S}_7, \mathscr{S}_8\}$, which is illustrated in Figure 4 by the gray and white area. If the resulting hinge point is out of a predefined area, e.g. outside the subgroups $\{\mathscr{S}_3, \dots, \mathscr{S}_{K-2}\}$, the balance of the groups is adjusted and two new LSR models are determined by the adjusted groups. Otherwise, the stopping criterion is fulfilled and the quality of the

splitting criterion, resulting from the candidate x_m and the hinge point as a threshold value [10], is measured.

The quality is measured by a specific criterion, which results in the quality value

$$\gamma = \frac{|\mathbf{e}_\mathrm{h}|}{|\mathbf{e}_\mathrm{lsr}|} \tag{7}$$

with the residuals $\mathbf{e}_\mathrm{h} = [e_1 - h(\hat{y}_{\mathrm{s}_k,1}) \ \ldots \ e_N - h(\hat{y}_{\mathrm{s}_k,N})]^\mathrm{T}$ of $h(\hat{y}_{\mathrm{s}_k})$. The quality criterion describes how much the non-linearity in a partition can be reduced along a certain direction by the splitting criterion. A decrease of non-linearity is indicated by $\gamma < 1$ and to fulfill the quality criterion within a forward iteration, the candidate x_m has to be selected in a way that γ is minimized to γ^*. Due to this minimization, the direction of non-linearity in a partition is identified which can most likely be approximated by two local linear models. Furthermore, this minimization effects that

- the orientation of $\hat{y}_{\mathrm{s}_k}(\mathbf{x}^*, x_m)$ becomes more similar to $\nabla f(\mathbf{x})$ in the area of curvature in that partition.
- the partition is split in an area near the curvature due to the hinge point.

Apart from these improvements, the computational effort to select a suitable splitting criterion increases linearly by $M \cdot \lambda$, whereby the curse of dimensionality is weakened.

If a new local optimal quality value is measured ($\gamma < \gamma^*$), in step c) of Figure 3 a new suitable splitting criterion is created. After the greedy search was applied ($m = M$) and no suitable candidate was identified ($\gamma \geq \gamma^*$), the whole FSM is stopped. Otherwise, $\mathbf{x}^*$ is extended by x_m that minimizes γ^* and, if complexity limitation isn't reached ($\dim(\mathbf{x}^*) < \lambda + 1$), the FSM is continued. The FSM is successfully completed if at least one suitable candidate has been identified. In this case $\boldsymbol{\beta}^*$ and c^* construct either an uni- or multivariate splitting criterion. If no input variables are selected by the FSM or if a minimal number of samples is reached, a local model for prediction is determined, which is described next.

3.3 Local Models

Similar to the requirements on the multivariate splitting criterion, the local models must be accurate, as interpretable as possible and well generalized. In order to achieve this, stepwise regression with another FSM is performed. Due to the bias-corrected *Akaike's Information Criterion*

$$\mathrm{AIC_C} = N \log \frac{\mathrm{RSS}}{N} + 2\tilde{M} + N + N\log(2\pi) + \frac{2(\tilde{M}+2)(\tilde{M}+3)}{N-(\tilde{M}+2)-1} \quad , \tag{8}$$

which is embedded into the FSM, both the model accuracy and complexity are taken into account during the forward selection [15]. The first term of (8) considers the model accuracy using the residual sum of squares

$$\mathrm{RSS} = \sum_{n=1}^{N} e_n^2 \tag{9}$$

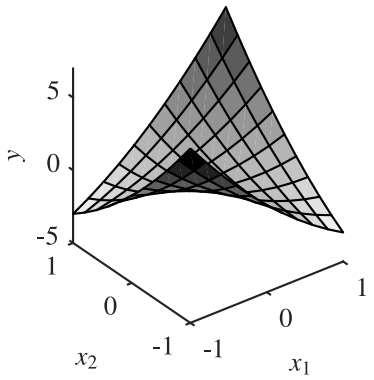
and the remaining terms are considering model complexity using the dimension of selected input variables $\tilde{M}$ and the number of samples N. $\mathrm{AIC_C}$ differs from the uncorrected criterion through the additional bias-correction term resulting from the last fraction, which leads to an improvement in accuracy for small data sets or high dimensional input spaces [15]. In this paper, the method is limited to a first-order LSR model

$$\hat{y}_{\tilde{\mathrm{I}}_k}(\tilde{\mathbf{x}}) = \beta_0 + \sum_{m \in \mathbb{N} \mid x_m \in \tilde{\mathbf{x}}} \beta_m x_m \quad , \tag{10}$$

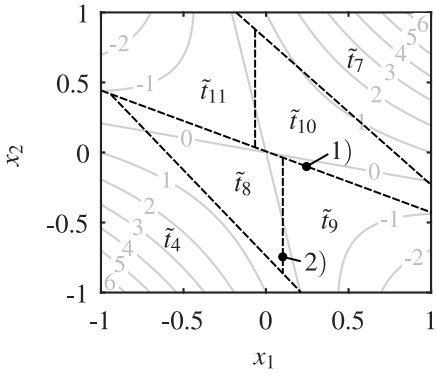
which is constructed by the selected input variables $\tilde{\mathbf{x}}$ with $\dim(\tilde{\mathbf{x}}) = \tilde{M}$. This is illustrated next using a practical example.

3.4 Tree Structure

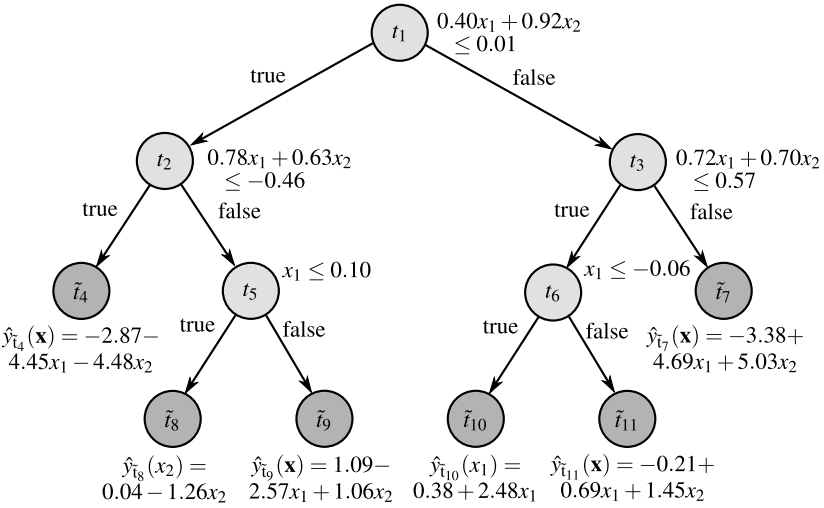
The proposed algorithm constructs a binary tree, called LSRT, using both uni- and multivariate splitting criteria. Figure 5c shows a *Mixed Regression Tree* that approximates the function $f(\mathbf{x}) = 5x_1x_2 + x_1^2 + x_2^2$ which is presented in Figure 5a.



(a) Test function $f(\mathbf{x}) = 5x_1x_2 + x_1^2 + x_2^2$ on which LSRT was trained using 50 samples generated from $f(\mathbf{x})$.



(b) Partitions resulting from LSRT and represented by $\tilde{t}_k$. The splits (black dashed lines) are adapted to functions' gray contour lines.



(c) LSRT consisting of uni- and multivariate splitting criteria (right to t_k) and local linear models $\hat{y}_{\tilde{t}_k}$ (below $\tilde{t}_k$).

Figure 5: *Mixed Regression Tree* which is called LSRT and constructed by the proposed algorithm.

The tree was trained on 50 samples and to cover the whole input space, these samples and generated from an optimized *Latin Hypercube Design* [16]. The tree consists of five decision nodes $t_k \; \forall \; k \in \{1,2,3,5,6\}$ with an uni- or multivariate splitting criterion displayed to the right of the node and six terminal leaves $\tilde{t}_k \; \forall \; k \in \{4,8,9,10,11,7\}$ with a local model $\hat{y}_{\tilde{t}_k}(\tilde{\mathbf{x}})$.

Figure 5b presents the partitions of the input space resulting from the splitting criteria, which are drawn by black dashed lines. For instance, the axis-oblique split 1) results from the multivariate splitting criterion of the root t_1 and the axis-orthogonal split 2) results from the univariate splitting criterion of node t_5. Each partition contains a first-order LSR model $\hat{y}_{\tilde{t}_k}(\tilde{\mathbf{x}})$, which is only valid in its partition. It can be recognized that the direction of the splits are adapted to the gray contour lines of $f(\mathbf{x})$, which indicates that the algorithm determines suitable splits. To investigate the performance of the algorithm in more detail, in the following an extensive experimental analysis is performed.

4 Experimental Analysis

In order to analyze the proposed algorithm with regard to accuracy and model complexity, the algorithm is tested in Subsection 4.1 on synthetic data and in Subsection 4.2 on real-world data. Furthermore, to compare the performance to state-of-the-art construction algorithms for *Regression Trees*, LSRT is compared to GUIDE, which is determined by a toolbox [17].

To obtain comparable results among LSRT and GUIDE, the trees are constructed based on similar hyperparamters. Both trees are pruned by the same method using the same hyperparameters and splitting is stopped by a lower bound of six samples. In addition, the local models are both determined using a FSM. To ensure the interpretability of LSRT, multivariate splitting criteria are limited to $\lambda = 2$. Although both trees are constructed on similar hyperparameters, the size of the pruned tree can vary between LSRT and GUIDE. For a comparison without the restriction of the tree size, a third tree LSRT_{adj} is considered that is pruned to the same size as GUIDE.

Test results are evaluated by the root mean squared error $\bar{E}$ and the tree size $|\bar{T}|$, measured by the number of nodes within the tree T. To generate meaningful results, $\bar{E}$ and $|\bar{T}|$ are averaged over 150 runs.

4.1 Synthetic Data

To generate synthetic data, a common test function from [18] is extended to

$$f(\mathbf{x}) = \sum_{i=1}^{I} \frac{10}{i^2} \sin(\pi x_{5i-4} x_{5i-3}) + \frac{20}{i^2}(x_{5i-2} - 0.5)^2 + \frac{10}{i^2} x_{5i-1} + \frac{5}{i^2} x_{5i} \ , \quad (11)$$

so that the dimensionality of $\mathbf{x} \in \mathbb{R}^M \mid M = I \cdot 5$ can be varied in discrete steps $I \in \mathbb{N}$ of five. In this way, the influence of dimensionality can be analyzed. The input space is limited to $0 \leq x_m \leq 1$ and the N samples for training are generated from an optimized *Latin Hypercube Design* [16] to fill the whole input space. To analyze the influence of noise, white Gaussian noise ε with mean $\bar{\varepsilon} = 0$ and variance $\sigma^2 = 0.5$ is added to $f(\mathbf{x})$. Furthermore, M_n noisy input variables without a dependence on $f(\mathbf{x})$ are constructed using ε with $\bar{\varepsilon} = 0.5$ and $\sigma^2 = 0.1$. In each of the 150 runs, 1500 samples are randomly generated from (11) for testing. Table 1 shows the experimental results on eight synthetic data sets.

For all data sets, LSRT and LSRT_{adj} are more accurate than GUIDE. Compared to GUIDE, the error of LSRT is reduced by 18.8% and the error of LSRT_{adj}, which has the same complexity as GUIDE, is reduced by 13.6%. Furthermore, it can be recognized that the difference in error between GUIDE and LSRT_{adj} is slight for data set $\{50,5,0,0\}$, whereas the difference for $\{300,5,0,0\}$ is significant (20.2%). These differences result from the mixed tree structure of LSRT. Due to the properties of $f(\mathbf{x})$, axis-oblique splits occur in deeper layers of LSRT. With an average tree size of $|\bar{T}| = 2.2$ in data set $\{50,5,0,0\}$, LSRT_{adj} consists only of univariate splits. In contrast, LSRT_{adj} with an average size of $|\bar{T}| = 15.6$ consists of several axis-oblique splits, which demonstrates the improvements resulting from the oblique splits.

Table 1 shows that an influence of noise can be handled well by the proposed algorithm. The results of LSRT for the noisy data sets $\{300,5,0,0.5\}$ and

Table 1: Experimental results on synthetic data for two different trees LSRT and GUIDE. LSRT_{abj} is an complexity adjusted version of LSRT with the same size as GUIDE. The elements in the brackets (left column) indicate the properties of the eight data sets. The best results for model complexity $|\bar{T}|$ and test error $\bar{E}$ are in bold print.

Settings	LSRT		GUIDE		LSRT_{adj}
$\{N, M, M_{\text{n}}, \sigma_\varepsilon^2\}$	$\lvert\bar{T}\rvert$	$\bar{E} \pm \sigma$	$\lvert\bar{T}\rvert$	$\bar{E} \pm \sigma$	$\bar{E} \pm \sigma$
$\{50,5,0,0\}$	3.9	2.13±0.19	2.2	2.35±0.20	2.31±0.30
$\{100,5,0,0\}$	7.1	1.73±0.27	4.1	2.07±0.16	1.94±0.20
$\{200,5,0,0\}$	11.0	1.23±0.14	11.6	1.52±0.15	1.23±0.17
$\{300,5,0,0\}$	11.4	1.09±0.26	15.6	1.19±0.26	0.95±0.14
$\{300,10,0,0\}$	11.0	1.42±0.17	8.2	1.92±0.32	1.68±0.32
$\{300,15,0,0\}$	9.5	1.67±0.19	4.1	2.17±0.13	2.08±0.21
$\{300,5,5,0\}$	11.6	1.11±0.18	13.1	1.64±0.41	1.24±0.36
$\{300,5,0,0.5\}$	11.9	1.10±0.23	13.8	1.30±0.22	1.03±0.13

$\{300,5,5,0\}$ are similar to the results of $\{300,5,0,0\}$. In addition, the error of LSRT resulting from $\{300,5,5,0\}$ is 32.3% lower than the error of GUIDE. An influence of dimensionality cannot be evaluated clearly. Due to an increase of the function values by the addition of further terms, $\bar{E}$ is equally increased. Based on the error reduction (23.0%) between LSRT and GUIDE for $\{300,15,0,0\}$, it can be expected that the curse of dimensions is weakened.

In four out of eight data sets, both LSRT and GUIDE have the lowest complexity, which means that both trees achieve comparable results in interpretability. A comparison between LSRT and LSRT_{adj} shows that for $\{300,5,0,0\}$ and $\{300,5,0,0.5\}$ tree size was penalized too much by the pruning method. This can be recognized by the lower test error of LSRT_{adj}. In the following, LSRT is tested in a more challenging task using real-world data.

4.2 Real-World Data

In contrast to synthetic data, real-world data provides more challenging tasks for data driven-models due to incomplete samples, outliers and skewed data. To analyze and compare the proposed algorithm with regard to a more challenging task, four different real-world data sets Baseball, Tecator, CPU and Redwine

Table 2: Experimental results on real-word data for two different trees LSRT and GUIDE. LSRT_{abj} is an complexity adjusted version of LSRT with the same size as GUIDE. The real-world data sets Baseball and CPU are scaled by 10^{-2} and 10^{-1}.

	LSRT		GUIDE		LSRT_{adj}
Data sets	$\lvert\bar{T}\rvert$	$\bar{E} \pm \sigma$	$\lvert\bar{T}\rvert$	$\bar{E} \pm \sigma$	$\bar{E} \pm \sigma$
Baseball	3.1	2.273±0.118	3.0	2.346±0.088	2.255±0.101
Tecator	3.6	0.903±0.057	1.6	0.926±0.065	0.858±0.040
CPU	5.2	5.214±0.800	4.3	5.129±0.514	5.471±1.157
Redwine	3.3	0.645±0.007	1.8	0.653±0.007	0.652±0.007

with a dimension from 6 to 24 input variables and a size from 209 to 1599 samples are used [19, 20, 21, 22]. Because of LSRTs' limitation to numerical input variables, categorical input variables are excluded from the data sets. In addition, the Tecator data set is reduced by 100 input variables containing the absorbance spectrum. Because of the small sample size of CPU with $N = 209$ and Tecator with $N = 240$ the analysis is performed by a k-fold cross validation. Within each run and each data set, k trees of each type are trained by $k-1$ varying data subsets, which predict the remaining data subset [9]. For this purpose, CPU and Tecator are analyzed by $k = 10$, Baseball by $k = 5$ and Redwine by $k = 2$. The results are shown in Table 2.

Compared to the results on synthetic data, the improvements by the proposed algorithm are less significant. On average, the error of LSRT is 1.3% and the error of LSRT_{abj} is 1.2% less than that of GUIDE. Furthermore, GUIDE is less complex for each data set. Nevertheless, due to a maximum size of $|\bar{T}| = 5.2$ there are no limitations in interpretability.

Improvements of an axis-oblique structure are only apparent at the Baseball data set. The root of LSRT is split by a multivariate criterion, which reduces the error versus GUIDE by 3.9%. For Tecator, the error of LSRT_{abj} is 7.3% less than the error of GUIDE. Because of the small tree size ($|\bar{T}| = 1.6$), the error reduction only result from the FSM to determine the local model. This can be explained by distinct linear dependencies, which can be well fitted with a single multiple regression model. A multivariate split of the root, which is performed by LSRT ($|\bar{T}| = 3.6$), results in an increase of error. Due to Tecators'

dimension of 24, this could be caused either by a wrong split selection or by a limitation $\lambda = 2$ of two input variables for splitting. Compared to GUIDE, LSRT performs slightly worse on the CPU data set. Five out of six input variables of CPU are discrete, so the segmentation process of the split selection (compare Subsection 3.2) does not work correctly anymore. The Redwine data set contains much noise and functional dependencies are low. Therefore, an increase in accuracy of LSRT may result from an increase in complexity.

5 Conclusion

To solve the issues of *Regression Trees* with respect to model accuracy, their structure can be extended to *Mixed* or *Oblique Regression Trees* using axis-oblique splits. In order to obtain the advantages of *Regression Trees* when using axis-oblique splits, a trade-off between an increase in model accuracy and a loss of interpretability must be found. In this paper, a novel construction algorithm for *Mixed* and *Oblique Regression Trees* was presented. The direction for splitting within a partition is determined by a first-order LSR model $\hat{y}_{s_k}(\mathbf{x}^*)$, which is limited to a maximum number of significant input variables $\mathbf{x}^*$ due to a forward selection method. Depending on the number of selected input variables, this direction can be either axis-orthogonal or axis-oblique. The selection of $\hat{y}_{s_k}(\mathbf{x}^*)$ is based on a quality criterion, which is determined by an approximation of candidate models' residuals using a hinge function. In this way, a split direction adapted to functions' curvature within the partition is obtained and the resulting one-dimensional search space for the selection weakens the curse of dimensionality. By the limitation to significant input variables, interpretability and generalization is maintained. The proposed algorithm was tested in an extensive experimental analysis using synthetic and real-world data and compared with a state-of-the-art algorithm for *Regression Trees*. Especially for synthetic data, significant improvements in model accuracy are achieved, resulting in lower test error compared to the state-of-the-art algorithm. The improvements for real-world data were less significant due to effects like discrete input values and partially unsuitable data sets. To obtain meaningful results on real-world data, further experiments are necessary.

For further improvements on real-world data, statistical test and an approach to consider categorical input variables and discrete values can be included into the split selection process. To improve the over-all performance of the proposed algorithm, the hinge function should be determined by a common algorithm in combination with a bootstrapping process. Stepwise selection combined with a complexity penalty for $\hat{y}_{s_k}(\mathbf{x}^*)$ could also provide further improvements in split selection. Furthermore, to increase model flexibility by curved splits, $\hat{y}_{s_k}(\mathbf{x}^*)$ could be extended to a higher order. Additionally, trees' structure can be extended to a neuro-fuzzy structure and to decrease the computational effort of the proposed algorithm, an efficient technique for prepruning is necessary.

Acknowledgements

This work was supported by the EFRE-NRW funding programme "Forschungsinfrastrukturen" (grant no. 34.EFRE-0300180).

References

[1] Brenno Menezes, Jeffrey Kelly, Adriano Leal and Galo Carrillo Le Roux. "Predictive, Prescriptive and Detective Analytics for Smart Manufacturing in the Information Age". In: *IFAC-PapersOnLine*, 52:568–573. 2019.

[2] Chao Shang and Fengqi You. "Data Analytics and Machine Learning for Smart Process Manufacturing: Recent Advances and Perspectives in the Big Data Era". In: *Engineering*, 5(6):1010 – 1016. 2019.

[3] Puran Tewari, Kapil Mittal and Dinesh Khanduja. "An Insight into Decision Tree Analysis". In: *World Wide Journal of Multidisciplinary Research and Development*, 3:111–115. 2017.

[4] Wei-Yin Loh. "Fifty Years of Classification and Regression Trees". In: *International Statistical Review*, 82. 2014.

[5] Lior Rokach and Oded Maimon. “Data Mining With Decision Trees: Theory and Applications”. World Scientific Publishing Co., Inc., USA, 2nd edition. 2014.

[6] W.-Y. Loh. “Regression Trees With Unbiased Variable Selection and Interaction Detection”. In: *Statistica Sinica*, 12:361–386. 2002.

[7] Carla E. Brodley and Paul E. Utgoff. “Multivariate Decision Trees”. In: *Machine Learning*, 19(1):45–77. 1995.

[8] Marek Kretowski. “Evolutionary Decision Trees in Large-Scale Data Mining”. Springer International Publishing. 2019.

[9] Oliver Nelles. “Nonlinear System Identification”. Springer Berlin Heidelberg. 2001.

[10] Ker-Chau Li, Heng-Hui Lue and Chun-houh Chen. “Interactive Tree-Structured Regression via Principal Hessian Directions”. In: *Journal of the American Statistical Association*, 95:547–560. 2000.

[11] P. Chaudhuri, M.-C. Huang, W.-Y. Loh and R. Yao. “Piecewise-Polynomial Regression Trees”. In: *Statistica Sinica*, 4:143–167. 1994.

[12] Leo Breiman, Jerome Friedman, Charles J. Stone and R.A. Olshen. “Classification and Regression Trees”. Chapman and Hall/CRC, New York. 1984.

[13] Ethem Alpaydin. “Introduction to Machine Learning”. The MIT Press, 2nd edition. 2010.

[14] Tamás Kenesei and János Abonyi. “Hinging hyperplane based regression tree identified by fuzzy clustering and its application”. In: *Applied Soft Computing*, 13(2):782–792. 2013.

[15] Charles Lindsey and Simon Sheather. “Variable Selection in Linear Regression”. In: *The Stata Journal*, 10(4):650–669. 2010.

[16] Tobias Ebert, Torsten Fischer, Julian Belz, Tim Oliver Heinz, Geritt Kampmann and Oliver Nelles. “Extended Deterministic Local Search Algorithm for Maximin Latin Hypercube Designs”. In: *IEEE Symposium Series on Computational Intelligence*. 2015.

[17] Wei-Yin Loh. “GUIDE Classification and Regression Trees and Forests (version 35.2)”, http://pages.stat.wisc.edu/ loh/guide.html, Last accessed 25 September 2020.

[18] Jerome H. Friedman, Eric Grosse and Werner Stuetzle. “Multidimensional Additive Spline Approximation”. In: *SIAM Journal on Scientific and Statistical Computing*, 4(2):291–301. 1983.

[19] Joaquin Vanschoren, “OpenML baseball-hitter”, https://www.openml.org/d/525, Last accessed 25 September 2020.

[20] Joaquin Vanschoren, “OpenML tecator”, https://www.openml.org/d/505, Last accessed 25 September 2020.

[21] Jan van Rijn. “OpenML machine_cpu”, https://www.openml.org/d/230, Last accessed 25 September 2020.

[22] Pieter Gijsbers, “OpenML wine-quality-red”, https://www.openml.org/d/40691, Last accessed 25 September 2020.