Machine Learning
https://doi.org/10.1007/s10994-021-06091-7

Coefficient tree regression: fast, accurate and interpretable predictive modeling

Özge Sürer[1] · Daniel W. Apley[1] · Edward C. Malthouse[1]

Received: 29 February 2020 / Revised: 17 May 2021 / Accepted: 29 September 2021

Abstract
The proliferation of data collection technologies often results in large data sets with many observations and many variables. In practice, highly relevant engineered features are often groups of predictors that share a common regression coefficient (i.e., the predictors in the group affect the response only via their collective sum), where the groups are unknown in advance and must be discovered from the data. We propose an algorithm called coefficient tree regression (CTR) to discover the group structure and fit the resulting regression model. In this regard CTR is an automated way of engineering new features, each of which is the collective sum of the predictors within each group. The algorithm can be used when the number of variables is larger than, or smaller than, the number of observations. Creating new features that affect the response in a similar manner improves predictive modeling, especially in domains where the relationships between predictors are not known a priori. CTR borrows computational strategies from both linear regression (fast model updating when adding/modifying a feature in the model) and regression trees (fast partitioning to form and split groups) to achieve outstanding computational and predictive performance. Finding features that represent hidden groups of predictors (i.e., a hidden ontology) that impact the response only via their sum also has major interpretability advantages, which we demonstrate with a real data example of predicting political affiliations with television viewing habits. In numerical comparisons over a variety of examples, we demonstrate that both computational expense and predictive performance are far superior to existing methods that create features as groups of predictors. Moreover, CTR has overall predictive performance that is comparable to or slightly better than the regular lasso method, which we include as a reference benchmark for comparison even though it is non-group-based, in addition to having substantial computational and interpretive advantages over lasso.

Editors: Tim Verdonck, Bart Baesens, María Óskarsdóttir and Seppe vanden Broucke.

✉ Özge Sürer
ozgesurer2019@u.northwestern.edu

Daniel W. Apley
apley@northwestern.edu

Edward C. Malthouse
ecm@northwestern.edu

[1] Industrial Engineering and Management Sciences, Northwestern University, Evanston, USA

Keywords Aggregation · Group structure · Ontology · Feature engineering

1 Introduction

Increasingly, phenomena occur in digital environments that can be recorded in great detail, producing data sets with a large number of predictors. For example, sensors track every movement of an object during manufacturing processes, medical devices such as pacemakers record every heartbeat, and smartphones record every user action or change in location. High dimensionality complicates the task of finding interpretable models with high predictive accuracy in a computationally efficient way (Fan and Li, 2006; Rudin, 2018). However, often a group of predictors all have the same effect on the response variable (i.e., they share a common coefficient in the regression model, which is equivalent to the predictors affecting the response only collectively via their sum), which enables a low-dimensional, group-based representation that overcomes many difficulties of high-dimensional analysis (Hastie et al., 2000; Tibshirani et al., 2005; Park et al., 2007). Since knowledge of the group structure is generally not available in advance, discovering them automatically from the data is fundamentally important. Finding such groups of predictors that share a common regression coefficient is a novel way of feature engineering, and is the focus of this article.

We consider the standard linear regression model with n observations and p predictors:

$$\mathbf{y} = \beta_1\mathbf{x}_1 + \beta_2\mathbf{x}_2 + \cdots + \beta_p\mathbf{x}_p + \boldsymbol{\epsilon}, \tag{1}$$

where $\mathbf{y} = \left[y_1, y_2, \ldots, y_n\right]^\top$, $\mathbf{x}_j = \left[x_{1,j}, x_{2,j}, \ldots, x_{n,j}\right]^\top$ and $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma^2\mathbf{I})$ are n dimensional vectors of the response observations, the predictor observations, and the noise, respectively, and each β_j is a regression coefficient. Without loss of generality we assume that each predictor and the response have been centered to have sample mean zero, so that there is no intercept in the model (1). We also assume that the predictors have been scaled in a meaningful way, per Remark 1 discussed later in Sect. 2.1.

As mentioned above, in high-dimensional problems with many predictors, we often expect the unknown coefficient vector to have a group structure. By group structure, we mean that there are groups of predictors, and the predictors within a group all have the same effect on the response (i.e., they share a common coefficient in (1)). If the predictors within a group share a common coefficient, this is equivalent to the group of predictors impacting the response only via a single *derived feature* that is the sum of the predictors in the group. In other words, letting G_i denote the set of indices in the ith group (for $i = 1, 2, \ldots, k$, where k is the number of distinct groups), and considering the derived feature $\mathbf{z}_i = \sum_{j \in G_i} \mathbf{x}_j$, the linear regression model in (1) can be written as

$$\mathbf{y} = \alpha_1\mathbf{z}_1 + \alpha_2\mathbf{z}_2 + \cdots + \alpha_k\mathbf{z}_k + \boldsymbol{\epsilon}, \tag{2}$$

where α_i is the common regression coefficient shared by all predictors in group i.

Applications of group-structured representations abound in many fields. For example, in an impact study designed to measure the impact of nutritional policies and environmental change on obesity in the high school presented by Huang et al. (2009), there are 25 predictors, and they naturally belong to groups that are known a priori. In this case, it is reasonable to assume that predictors in the same group share the same coefficients (or very nearly so). To illustrate this, consider the high-school students' consumption on "fizzy drinks", "sweets", "crisps", "cake" and "ice cream". When building a regression model to predict the body mass index of a student, instead of including the amount of consumption of "fizzy drinks", "sweets", "crisps", "cake" and "ice cream" as separate predictors, it may be preferable to construct a derived feature as the sum of their consumptions, which could be interpreted as the "unhealthy food consumption" group. In this case, letting $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, $\mathbf{x}_4$, and $\mathbf{x}_5$ denote the amount of "fizzy drinks", "sweets", "crisps", "cake" and "ice cream" consumption, respectively, a derived feature $\mathbf{z}_1 = \mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3 + \mathbf{x}_4 + \mathbf{x}_5$ represents "unhealthy food consumption". However, in many real problems we would not know in advance whether 'fizzy drinks," "sweets", "crisps", "cake" and "ice cream" should be grouped or kept as separate predictors. This renders methods that assume a known group structure inappropriate and requires that the structure be discovered empirically, from the data.

Apart from examples grouping continuous predictors, categorical predictors can often be split into the groups as well to obtain interpretable models (Carrizosa et al., 2017). As an illustration, consider the German credit data set `german` from the UCI repository including 11 categorical predictors. In this case, the total number of predictors increases from 11 to 52 by creating multiple dummy predictors out of a single categorical predictor. However, by grouping various categories of each categorical predictor (a derived feature representing a group of categories can be represented by the sum of the one-hot-encoded predictors for the individual categories), the total number predictors in the final model can be decreased. For example, Carrizosa et al. (2017) obtain the two groups {"real estate", "building society savings agreement/life insurance", "car or other"} and {"unknown/no property"} for the four categories of a categorical predictor "Property."

The aforementioned examples suggest that group-structured representations have widespread applicability in different fields. Finding the unknown group structure leads to simple, parsimonious and interpretable models. In this paper we develop an approach that iteratively identifies the unknown group structure by constructing derived features as the sum of the predictors in each group and then computes the estimates of the coefficients of the derived features per (2), all in a highly computationally efficient manner to be described. In this regard our approach is a new feature engineering technique that improves the predictive modeling because the feature construction process is integrated with the predictive modeling, and the derived features are constructed based on the similarity of their effects on the response variable.

There are a variety of data pre-processing approaches for using unsupervised learning to construct interpretable features in regression problems. Methods such as clustering provide insight into relationships contained within the unlabelled data. Dimensionality reduction focuses on finding a representation of the data in a low-dimensional space. Methods such as principal components analysis (Jolliffe, 1986), independent components analysis (Comon, 1992), canonical correlation analysis (Hotelling 1936) and factor analysis (Akaike, 1987; Fokoué and Titterington, 2003) can often identify a few interpretable dimensions. However, none of these methods consider the response variable when generating low-dimensional representation of the data. In this paper, we focus on a new method that derives new features while simultaneously fitting the predictive relationship between the response variable and the predictors in a linear regression model.

Group-structured representations in regression have been studied in the prior literature. We categorize existing methods by whether or not they assume prior knowledge about the group structure. The first category of methods assumes known groups (or known ordering of the coefficients). The group lasso and the fused lasso are the most common methods in this category. The group lasso selects important groups of predictors rather than individual ones, where all predictors are structured into groups that are known *a priori* (Yuan and Lin, 2006; Zhao et al., 2009), whereas the fused lasso penalizes the successive differences of coefficients assuming that their ordering is known *a priori* (Tibshirani et al., 2005). See group SCAD (Wang et al., 2007), group MCP (Breheny, 2009), group bridge (Huang et al., 2009), group hierarchical lasso (Zhou and Zhu, 2010), group exponential lasso (Breheny, 2015) for various regression and classification models with different penalty structures. Qiu and Ahn (2020) introduce new screening methods for previously grouped predictors to reduce the number of groups and the predictors within those groups.

The focus of this paper is on the second category of methods, which assumes no prior knowledge of the group structure, so that the structure must be discovered from the same data to which the model is fit. If one had knowledge of the group structure in advance, then we would expect that an approach that takes this known structure into account would be more effective than an approach that does not. However, an unknown group structure is much more widely applicable in high-dimensional problems involving predictive relationships that are not well understood in advance, and so computationally efficient methods are required for problems with unknown group structure. Dettling and Bühlmann (2004) propose the method *Pelora* to iteratively construct groups of genes that are strongly predictive of the response variable. Pelora fits a logistic regression model each time a group of predictors is modified. As in many other application areas such as biostatistics, predictive models that use a sum of predictors as a derived feature can perform better than more sophisticated methods, especially for large-p situations. To select a subset of useful predictors, Donoho and Jin (2008) and Zhao et al. (2014) allow a single derived feature that is the sum of selected predictors, where each predictor can have either a positive or negative sign in the summation. Since they only allow a single derived feature, the coefficient for every predictor must have the same magnitude. As in the case of the examples presented in these papers, often a formula that combines predictors with equal weights is as accurate as ones with unequal weights, while also being more robust to data quality issues and easier to interpret. The method OSCAR (Bondell and Reich, 2008) uses a penalty term for each pair of coefficients to encourage coefficients to be equal, whereas grouping pursuit (Shen and Huang, 2010; Zhu et al., 2013) considers non-convex penalties by shrinking only small differences of absolute values of coefficients. Ke et al. (2015) propose the method CARDS using within and between group penalties, which requires a preliminary ordering of coefficients to determine the group structure. These methods have high computational expense, which can be prohibitive with high-dimensional data.

In this paper, we propose a new algorithm that takes advantages of certain computational properties of both linear regression and regression trees in the following manner. Our aim is to find the groups by recursively splitting the predictors into sets that have similar coefficients, where each successive split is chosen to maximize the reduction in the sum of squared errors (SSE). Since an SSE optimization over the space of all splits is not computationally feasible, we use an iterative greedy search, analogous to how splits are chosen when fitting standard regression trees (but different in that our algorithm splits a set of predictors into two subgroups of predictors, whereas regression trees split an interval of values of a single predictor into two subintervals). To accelerate the splitting, we borrow a computational property from linear regression, whereby the reduction in the SSE when adding a new feature to the model is efficiently computed. The above computational properties result in an algorithm that is multiple orders-of-magnitude faster than existing group-based regression methods, in addition to having far superior predictive performance (see Sect. 3). Our approach even has nearly an order-of-magnitude better computational expense than fast state-of-the-art implementations (`glmnet`, Friedman et al. (2010)) of the popular lasso method, in addition to having substantial interpretability advantages.

Our approach results in an interpretable tree structure representing the groups of predictors and their associated coefficients (this includes the final group structure, as well as the sequence of splits that produced the group structure, which contains information on higher-level group structures). Hence, we call the approach coefficient tree regression (CTR). The tree structure enhances interpretability and provides insight into the engineered features and the hidden ontology of the predictors, which we demonstrate with a real data example of predicting political affiliations with television viewing habits in Sect. 4. We built the `CTR` R package by implementing the efficient computational properties to be described, and it also can be used as a visualization tool to provide insight into the regression dependencies.

The remainder of the paper is organized as follows. The proposed method and computational issues are discussed in Sect. 2. Sections 3 and 4 present the simulation studies and the results of a real data analysis, respectively. Section 5 describes how CTR can be viewed as a much faster and more accurate alternative to OLS. Finally, some concluding remarks are given in Sect. 6.

2 Coefficient tree regression (CTR)

2.1 Overview of the CTR model building procedure

Our approach iteratively identifies groups of predictors that share common coefficients, where the sum of predictors in a group corresponds to a derived feature per model (2). During each iteration ($k = 1, 2, \ldots$), either a new group of predictors enters the model, or an existing one is split into two groups, in a manner that most reduces the SSE. During the process of updating the group structure at each iteration, CTR also fits a linear regression model with all derived features. This process continues until we obtain a final set of derived features. The total number of derived features in the final model is the only tuning parameter for CTR, and we select this via cross validation (CV), which will be described in Sect. 2.6.

We first introduce some notation. We refer to the set of derived features in the model as the "basis." For the sake of notational clarity, we add a subscript k to each group G_i and the corresponding derived feature $\mathbf{z}_i$ produced after iteration k, at which point in the algorithm we always have k derived features present in the model. Then, letting $\mathbf{X}$ be the $n \times p$ design matrix $\mathbf{X} = \left[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p\right]$, a derived feature $\mathbf{z}_{k,i}$ becomes $\mathbf{z}_{k,i} = \sum_{j \in G_{k,i}} \mathbf{x}_j$. The fitted model with k derived features has the following structure:

$$\hat{\mathbf{y}}(\hat{\boldsymbol{\alpha}}_k) = \mathbf{Z}_k \hat{\boldsymbol{\alpha}}_k = \hat{\alpha}_{k,1}\mathbf{z}_{k,1} + \hat{\alpha}_{k,2}\mathbf{z}_{k,2} + \cdots + \hat{\alpha}_{k,k}\mathbf{z}_{k,k}, \tag{3}$$

where $\mathbf{Z}_k$ is an $n \times k$ matrix of derived features, $\hat{\boldsymbol{\alpha}}_k = \left[\hat{\alpha}_{k,1}, \hat{\alpha}_{k,2}, \ldots, \hat{\alpha}_{k,k}\right]^{\mathsf{T}}$ is the estimated coefficient vector, and $\hat{\mathbf{y}}(\hat{\boldsymbol{\alpha}}_k)$ is the fitted response vector as a function of $\hat{\boldsymbol{\alpha}}_k$. We also assume that the predictors are scaled in a meaningful way as stated in Remark 1.[1]

[1] **Remark 1:** CTR is not scale-independent since the group structure depends on the scaling of the predictors. If the predictors are originally in different units, one may want to scale them so as to make it as likely as possible that predictors within groups share the same (or similar) coefficient. However, as discussed in Sect. 5, CTR can still result in a model with good predictive accuracy even when there is no group structure and the coefficients vary continuously (although the group interpretability is less meaningful in this case), so that scaling to create a group structure is not necessary.

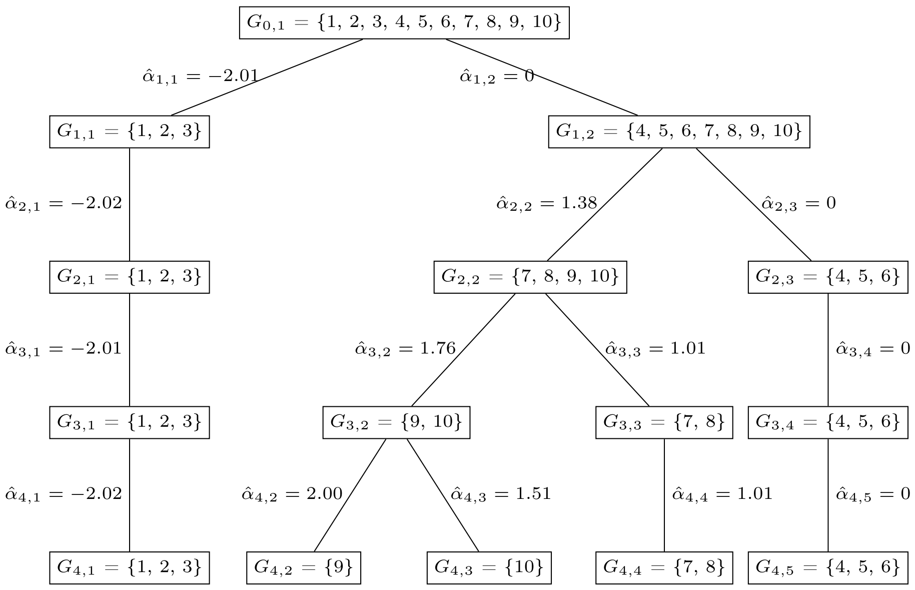

$G_{0,1} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$
$\hat{\alpha}_{1,1} = -2.01$
$\hat{\alpha}_{1,2} = 0$
$G_{1,1} = \{1, 2, 3\}$
$G_{1,2} = \{4, 5, 6, 7, 8, 9, 10\}$
$\hat{\alpha}_{2,1} = -2.02$
$\hat{\alpha}_{2,2} = 1.38$
$\hat{\alpha}_{2,3} = 0$
$G_{2,1} = \{1, 2, 3\}$
$G_{2,2} = \{7, 8, 9, 10\}$
$G_{2,3} = \{4, 5, 6\}$
$\hat{\alpha}_{3,1} = -2.01$
$\hat{\alpha}_{3,2} = 1.76$
$\hat{\alpha}_{3,3} = 1.01$
$\hat{\alpha}_{3,4} = 0$
$G_{3,1} = \{1, 2, 3\}$
$G_{3,2} = \{9, 10\}$
$G_{3,3} = \{7, 8\}$
$G_{3,4} = \{4, 5, 6\}$
$\hat{\alpha}_{4,1} = -2.02$
$\hat{\alpha}_{4,2} = 2.00$
$\hat{\alpha}_{4,3} = 1.51$
$\hat{\alpha}_{4,4} = 1.01$
$\hat{\alpha}_{4,5} = 0$
$G_{4,1} = \{1, 2, 3\}$
$G_{4,2} = \{9\}$
$G_{4,3} = \{10\}$
$G_{4,4} = \{7, 8\}$
$G_{4,5} = \{4, 5, 6\}$

Fig. 1 Illustration of a group structure produced in the CTR tree growing procedure with $p = 10$ predictors and 4 groups in the final model

CTR represents and graphically depicts the estimated model with an interpretable tree structure, as illustrated in Fig. 1. In the tree, each node corresponds to a group (i.e., a set of indices of a group of predictors that defines one derived feature per model (2)), and the label on the branch just above the node represents the estimated coefficient of the corresponding derived feature. Let $G_{k,1:k} = G_{k,1} \cup G_{k,2} \cup \cdots \cup G_{k,k}$ denote the indices of all predictors in all groups in the model after iteration k and $G_{k,k+1} = \{1, 2, \ldots, p\} \setminus G_{k,1:k}$ denote the excluded group of predictors after iteration k (which is equivalent to group $G_{k,k+1}$ having a coefficient of zero). At iteration $k = 0$, we start at the root node with all predictors in the excluded group, i.e., $G_{0,1} = \{1, \ldots, p\}$, and at each iteration k we grow the tree by splitting one of the current groups into two groups. The existing groups that were not split are carried over to the next iteration without modification. Thus, the number of groups increases by one during each iteration. After iteration k, we have $k + 1$ disjoint groups such that $G_{k,1} \cup G_{k,2} \cup \cdots \cup G_{k,k+1} = \{1, \ldots, p\}$ and $G_{k,i} \cap G_{k,j} = \emptyset \quad \forall \{i,j\} \subseteq \{1, \ldots, k+1\}$. During iteration $k + 1$, one of the $k + 1$ groups produced after iteration k is split into two groups, and then the coefficients for all derived features are updated. The tree is grown until we reach some terminal level of groups based on a stopping criterion that we describe in Sect. 2.6. The set $\{G_{k,1}, \ldots, G_{k,k}\}$ at the terminal iteration k comprises the groups of predictors in the final model.

During each iteration k, CTR must decide which group of predictors to split (forming a new group of predictors from previously excluded ones can be viewed as splitting the current group $G_{k-1,k}$ of zero-coefficient predictors into a smaller zero-coefficient group and a nonzero-coefficient group) and which predictors within the split group should be placed into each of the two subgroups. CTR chooses the group to split by, for each group in $\{G_{k-1,i} : i = 1, \ldots, k\}$, finding the (approximately) best way to split it into two groups to minimize the SSE, and then selecting the group whose split most reduces the SSE. For a given group, finding the optimal split that exactly minimize the SSE is a combinatorial optimization problem that is generally computationally infeasible. For example, a group with 21 predictors can be split into two groups in $2^{20} - 1$ possible ways. Hence, we use an iterative greedy procedure that splits a given group of predictors into two disjoint groups in a highly efficient way to avoid the computationally prohibitive (virtually impossible) consideration of all possible splits. Our efficient split procedure depends on the predictors' individual effect on the response variable as described later in detail, and for a given group the total number of possible splits to be considered is at most the size of the corresponding group.

Note that during iteration $k = 1$, the root node with all predictors in the excluded group is split into two groups, which results in a model with a single derived feature. In this sense, the first iteration of CTR is similar to the method presented by Donoho and Jin (2008) and Zhao et al. (2014), which uses the sum of selected predictors as a single derived feature, except that predictors can have either positive or negative signs in the summation. In Donoho and Jin (2008), the predictors are selected based on the absolute value of their univariate t-score, and once included into the classification model, they are only allowed to have coefficients of either $+1$ or -1 depending on the signs of the predictors' t-scores. Similar to Donoho and Jin (2008), Zhao et al. (2014) assigns coefficients of either $+1$ or -1 based on the sign of the marginal estimates of the regression coefficients. In these studies, predictors are not allowed to have coefficients of different sizes. CTR on the other hand adjusts the coefficients through splitting groups of predictors iteratively.

Sections 2.2–2.6 describe details of the CTR algorithm. Section 2.2 illustrates the CTR model building procedure via a toy example. Section 2.3 describes how, for any given split of a group of predictors into two groups, the reduction in SSE can be very efficiently computed. Section 2.4 describes our greedy heuristic algorithm for efficiently searching for the best way to split each group into two groups. Section 2.5 explains how to update the model at the end of each iteration to reduce the computational expense. Section 2.6 describes the termination criterion for deciding when no further splits should be made.

2.2 Illustration of the CTR model building procedure

We now describe the example in Fig. 1 to illustrate the CTR tree growing procedure. The details of how the groups are chosen during each iteration (based on the data, with no prior knowledge of the group structure) are given in Sect. 2.4. In Fig. 1, each level corresponds to an iteration k and shows the groups produced after that iteration, and the rightmost group in the level is the zero-coefficient group $G_{k,k+1}$. For illustrative clarity, we use a small $p = 10$ and $n = 10^4$, and the data set is generated based on the linear model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma^2\mathbf{I})$ and $\boldsymbol{\beta} = \left[-2, -2, -2, 0, 0, 0, 1, 1, 2, 1.5\right]^{\mathsf{T}}$. Each regression observation (aka "row") $\left[x_{i,1}, x_{i,2}, \dots, x_{i,10}\right]^{\mathsf{T}}$ is sampled from a multivariate normal distribution with mean $\mathbf{0}$ and covariance $\boldsymbol{\Sigma} = \mathbf{I}$. σ^2 was computed via $\boldsymbol{\beta}^{\mathsf{T}}\boldsymbol{\Sigma}\boldsymbol{\beta}(\frac{1}{r^2} - 1)$ to achieve a true $r^2 = 0.9$ where $r^2 = \frac{\mathbb{V}(\mathbf{y}-\boldsymbol{\epsilon})}{\mathbb{V}(\mathbf{y})}$. There are four true groups of predictors with nonzero coefficients, which, at a higher level, can roughly be divided into a group of predictors with negative coefficients $(\mathbf{x}_1, \dots, \mathbf{x}_3)$ and a (nonhomogeneous) group with positive coefficients $(\mathbf{x}_7, \dots, \mathbf{x}_{10})$. We also included predictors with coefficients equal to zero to show how CTR performs variable selection. By keeping track of zero-coefficient predictors in the rightmost group in each level, we can observe the variable selection procedure step-by-step.

For initialization, since CTR has not identified any derived feature yet, we have only the single group $G_{0,1}$, which contains all predictors and assigns them a common coefficient of zero. During iteration $k = 1$, CTR split the initial group $G_{0,1}$ into two groups, where

$G_{1,1} = \{1, 2, 3\}$ corresponds to the group of predictors in the basis, and $G_{1,2} = \{4, 5, \dots, 10\}$ is the excluded group. After iteration $k = 1$, the model is

$$\hat{\mathbf{y}}(\hat{\boldsymbol{\alpha}}_1) = \hat{\alpha}_{1,1}\mathbf{z}_{1,1} = \hat{\alpha}_{1,1}(\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3), \tag{4}$$

with $\hat{\alpha}_{1,1} = -2.01$. Thus, the first derived feature $\mathbf{z}_{1,1}$ chosen by CTR represents the group of predictors having a negative relation with the response. During iteration $k = 2$, CTR found a new group of predictors by splitting the zero-coefficient group $G_{1,2}$ into a nonzero-coefficient group $G_{2,2} = \{7, 8, 9, 10\}$ and a zero-coefficient group $G_{2,3} = \{4, 5, 6\}$, so that the fitted model after iteration $k = 2$ is

$$\begin{aligned}\hat{\mathbf{y}}(\hat{\boldsymbol{\alpha}}_2) &= \hat{\alpha}_{2,1}\mathbf{z}_{2,1} + \hat{\alpha}_{2,2}\mathbf{z}_{2,2} \\ &= \hat{\alpha}_{2,1}(\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3) + \hat{\alpha}_{2,2}(\mathbf{x}_7 + \mathbf{x}_8 + \mathbf{x}_9 + \mathbf{x}_{10}),\end{aligned} \tag{5}$$

with $\hat{\alpha}_{2,1} = -2.02$, $\hat{\alpha}_{2,2} = 1.38$. Thus, after iteration $k = 2$, we have a new group of predictors $G_{2,2}$ representing all predictors having a positive effect on the response, together with the previous group $G_{2,1}$ of predictors that have a negative effect on the response.

Within an existing group of predictors, if some are more or less influential than others in this group, the CTR algorithm can split them into separate subgroups at subsequent iterations to properly adjust their coefficients. For example, since the true coefficients of $\mathbf{x}_9$ and $\mathbf{x}_{10}$ are larger than the true coefficients of $\mathbf{x}_7$ and $\mathbf{x}_8$, during iteration $k = 3$ CTR split the group $G_{2,2} = \{7, 8, 9, 10\}$ into two subgroups $G_{3,2} = \{9, 10\}$ and $G_{3,3} = \{7, 8\}$. Thus, after iteration $k = 3$, the fitted model is

$$\begin{aligned}\hat{\mathbf{y}}(\hat{\boldsymbol{\alpha}}_3) &= \hat{\alpha}_{3,1}\mathbf{z}_{3,1} + \hat{\alpha}_{3,2}\mathbf{z}_{3,2} + \hat{\alpha}_{3,3}\mathbf{z}_{3,3} \\ &= \hat{\alpha}_{3,1}(\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3) + \hat{\alpha}_{3,2}(\mathbf{x}_9 + \mathbf{x}_{10}) + \hat{\alpha}_{3,3}(\mathbf{x}_7 + \mathbf{x}_8),\end{aligned} \tag{6}$$

with $\hat{\alpha}_{3,1} = -2.01$, $\hat{\alpha}_{3,2} = 1.76$, $\hat{\alpha}_{3,3} = 1.01$. During iteration $k = 4$, the group $G_{3,2} = \{9, 10\}$ was split into two subgroups, after which the fitted model was

$$\begin{aligned}\hat{\mathbf{y}}(\hat{\boldsymbol{\alpha}}_4) &= \hat{\alpha}_{4,1}\mathbf{z}_{4,1} + \hat{\alpha}_{4,2}\mathbf{z}_{4,2} + \hat{\alpha}_{4,3}\mathbf{z}_{4,3} + \hat{\alpha}_{4,4}\mathbf{z}_{4,4} \\ &= \hat{\alpha}_{4,1}(\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3) + \hat{\alpha}_{4,2}\mathbf{x}_9 + \hat{\alpha}_{4,3}\mathbf{x}_{10} + \hat{\alpha}_{4,4}(\mathbf{x}_7 + \mathbf{x}_8),\end{aligned} \tag{7}$$

with $\hat{\alpha}_{4,1} = -2.02$, $\hat{\alpha}_{4,2} = 2.00$, $\hat{\alpha}_{4,3} = 1.51$, $\hat{\alpha}_{4,4} = 1.01$. The total number of derived features in the final model is determined to be four via the 10-fold CV described in Sect. 2.6, and thus the final fitted model was (7). In Fig. 2, the CV SSE is given for different number of derived features in the final model ranging from 1 to 10, and the details regarding the selection of the best number of groups at termination are provided in Sect. 2.6. In this final model, the predictors with coefficients of zero in the true model were correctly excluded from the fitted model and remained in the final zero-coefficient group $G_{4,5} = \{4, 5, 6\}$. Note that the number of estimated coefficients decreases from 10 in the original regression model to four in the CTR model. In this sense, CTR encourages parsimonious models by grouping the predictors. Moreover, after the group structure is decided, one could fit a regression model with these four derived features and compute confidence intervals on the coefficients in the standard way to make a statistical inference. However, the confidence intervals would be distorted similarly to how they are distorted if the standard stepwise regression is used to select the predictor variables. The act of first searching over the data to select predictor variables or to select features composed of predictor variables invalidates the exactness of the specified confidence level (or the coverage probability) of the confidence intervals.

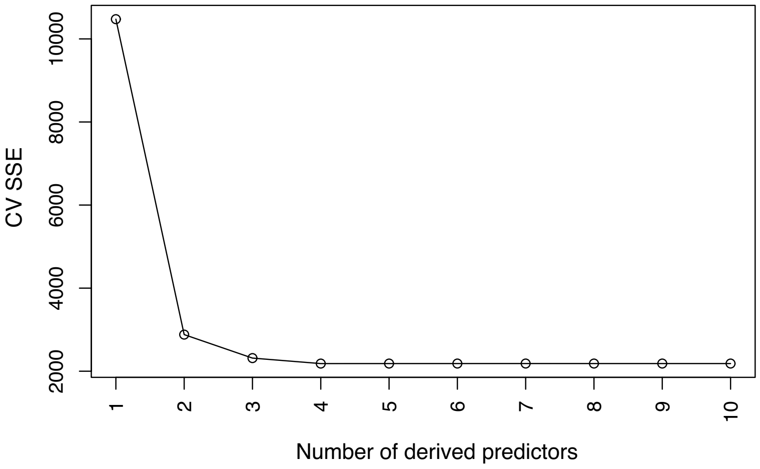


Fig. 2 CV SSE for the toy example in Fig. 1 with the the number of derived features in the final model ranging from 1 to 10.

2.3 Finding the reduction in SSE for a given split of a group

At iteration k, the goal is to split one of the existing groups $\{G_{k-1,i} : i = 1, 2, \dots, k\}$ into two groups in a manner that most reduces the SSE. Even with the heuristic search procedure described in the next section, the number of possible splits that must be considered is large. Hence, an efficient algorithm for computing the reduction in SSE for each given split is essential. Towards this end, consider that the basis after iteration k (i.e., after one of the existing groups is split into two) can be represented as follows, making use of the existing basis. At the end of iteration $k-1$, we have the existing derived features $\mathbf{Z}_{k-1}$. If the group that is split during iteration k is $G_{k-1,k}$ (i.e., a new group is added from the excluded predictors), then the basis after iteration k is $\mathbf{Z}_k = \left[\mathbf{Z}_{k-1}, \mathbf{z}\right]$ for a single new derived feature $\mathbf{z} = \mathbf{z}_{k,k}$.

Suppose instead that one of the existing groups of included predictors $\{G_{k-1,i} : i = 1, 2, \dots, k-1\}$ is split during iteration k and (without loss of generality) the groups are reordered so that the split group is $G_{k-1,k-1}$. On the surface, splitting one of the existing groups actually creates two new derived features and removes one existing derived feature in the basis. In this case, the $k-2$ predictors $\left[\mathbf{z}_{k-1,1}, \mathbf{z}_{k-1,2}, \cdots, \mathbf{z}_{k-1,k-2}\right]$ remain unchanged during iteration k, so that the derived features after iteration k are $\mathbf{Z}_k = \left[\mathbf{z}_{k-1,1}, \mathbf{z}_{k-1,2}, \cdots, \mathbf{z}_{k-1,k-2}, \mathbf{z}_{k,k-1}, \mathbf{z}_{k,k}\right]$, whereas the derived features after iteration $k-1$ are $\mathbf{Z}_{k-1} = \left[\mathbf{z}_{k-1,1}, \mathbf{z}_{k-1,2}, \cdots, \mathbf{z}_{k-1,k-1}\right]$. However, the three sets $\left[\mathbf{z}_{k,k-1}, \mathbf{z}_{k,k}\right]$, $\left[\mathbf{z}_{k-1,k-1}, \mathbf{z}_{k,k-1}\right]$, and $\left[\mathbf{z}_{k-1,k-1}, \mathbf{z}_{k,k}\right]$ all span the same two-dimensional subspace, because $\mathbf{z}_{k,k-1}$ and $\mathbf{z}_{k,k}$ were both formed by a single split of $\mathbf{z}_{k-1,k-1}$, i.e., $\mathbf{z}_{k-1,k-1} = \mathbf{z}_{k,k-1} + \mathbf{z}_{k,k}$. Thus, $\mathbf{Z}_k$ and $\left[\mathbf{Z}_{k-1}, \mathbf{z}\right]$ span the same k-dimensional subspace for either $\mathbf{z} = \mathbf{z}_{k,k-1}$ or $\mathbf{z} = \mathbf{z}_{k,k}$, and the SSE after splitting an existing group $G_{k-1,k-1}$ is exactly the same as the SSE after adding a single new derived feature $\mathbf{z}$ ($= \mathbf{z}_{k,k-1}$ or $\mathbf{z}_{k,k}$) to the basis $\mathbf{Z}_{k-1}$. Consequently, splitting an existing group into two subgroups also increases the dimension of the basis by only one. To illustrate this, consider the toy example presented in Fig. 1. During iteration $k = 3$, group $G_{2,2}$ is split into two subgroups such that the corresponding derived features are $\mathbf{z}_{2,2} = \mathbf{z}_{3,2} + \mathbf{z}_{3,3}$, where $\mathbf{z}_{2,2} = \mathbf{x}_7 + \mathbf{x}_8 + \mathbf{x}_9 + \mathbf{x}_{10}$, $\mathbf{z}_{3,2} = \mathbf{x}_9 + \mathbf{x}_{10}$ and $\mathbf{z}_{3,3} = \mathbf{x}_7 + \mathbf{x}_8$. But the two vectors $\{\mathbf{z}_{3,2}, \mathbf{z}_{3,3}\}$ clearly span the same subspace as $\{\mathbf{z}_{3,2}, \mathbf{z}_{2,2}\} = \{\mathbf{z}_{3,2}, \mathbf{z}_{3,2} + \mathbf{z}_{3,3}\}$, which span the same subspace as $\{\mathbf{z}_{3,3}, \mathbf{z}_{2,2}\} = \{\mathbf{z}_{3,3}, \mathbf{z}_{3,2} + \mathbf{z}_{3,3}\}$. Since all three of these pairs of vectors span the same subspaces, the SSEs for the regressions onto them are the same.

We first introduce some notation to compute the reduction in SSE. Let $\mathbf{P}_{k-1} \equiv \left[\mathbf{Z}_{k-1}\left(\mathbf{Z}_{k-1}^{\mathsf{T}}\mathbf{Z}_{k-1}\right)^{-1}\mathbf{Z}_{k-1}^{\mathsf{T}}\right]$ be the projection matrix onto the span of the derived features $\mathbf{Z}_{k-1}$ at the end of iteration $k-1$. For any n-length vector $\mathbf{w}$, let $\mathbf{e}_{\mathbf{w},k-1} = [\mathbf{I} - \mathbf{P}_{k-1}]\mathbf{w}$ denote the error in projecting $\mathbf{w}$ onto the span of $\mathbf{Z}_{k-1}$. In order to compute the reduction in SSE when the basis grows from $\mathbf{Z}_{k-1}$ to $\left[\mathbf{Z}_{k-1}, \mathbf{z}\right]$ (for any $\mathbf{z}$) in a computationally efficient manner, we use well-known geometric arguments for least squares with orthogonal projections. That is, $\left[\mathbf{Z}_{k-1}, \mathbf{z}\right]$ span the same space as $\left[\mathbf{Z}_{k-1}, \mathbf{e}_{\mathbf{z}}\right]$, where $\mathbf{e}_{\mathbf{z}}$ is the error in projecting $\mathbf{z}$ onto the span of $\mathbf{Z}_{k-1}$. Since $\mathbf{e}_{\mathbf{z}}$ is orthogonal to $\mathbf{Z}_{k-1}$ by construction, the reduction in the SSE when adding $\mathbf{z}$ (or $\mathbf{e}_{\mathbf{z}}$) to the basis is the regression sum of squares (SSR) for regressing $\mathbf{e}_{\mathbf{y}}$ onto the single derived feature $\mathbf{e}_{\mathbf{z}}$, where $\mathbf{e}_{\mathbf{y}}$ is the error in projecting $\mathbf{y}$ onto the span of $\mathbf{Z}_{k-1}$. Consequently, letting $R(\mathbf{z})$ denote the reduction in SSE with the addition of $\mathbf{z}$ into the basis, we have

$$R(\mathbf{z}) = \frac{\left(\mathbf{e}_{\mathbf{y},k-1}^{\mathsf{T}}\mathbf{e}_{\mathbf{z},k-1}\right)^2}{\mathbf{e}_{\mathbf{z},k-1}^{\mathsf{T}}\mathbf{e}_{\mathbf{z},k-1}} = \frac{N^2}{D}, \tag{8}$$

where $N \equiv \mathbf{e}_{\mathbf{z},k-1}^{\mathsf{T}}\mathbf{e}_{\mathbf{y},k-1}$ and $D \equiv \mathbf{e}_{\mathbf{z},k-1}^{\mathsf{T}}\mathbf{e}_{\mathbf{z},k-1}$.

2.4 The split-point search procedure

Algorithm 1: Pseudo-code for CTR

```
1  Initialize: $\mathbf{E}_0 = \mathbf{X} = [\mathbf{e}_{1,0}, \mathbf{e}_{2,0}, \ldots, \mathbf{e}_{p,0}]$, $(\mathbf{w})_{j,0} = \mathbf{e}_{j,0}^{\mathsf{T}}\mathbf{y}$, $(\mathbf{u})_{j,0} = \mathbf{e}_{j,0}^{\mathsf{T}}\mathbf{e}_{j,0}$ and
     $\mathbf{q} = [(\mathbf{q})_1, (\mathbf{q})_2, \ldots, (\mathbf{q})_p] = \mathbf{0}_{p\times 1}$
2  for $k = 1, \ldots, k_{max}$ do
3      Compute the $p \times 1$ vector $\mathbf{r}$, the $j$th elements of which are
4      $(\mathbf{r})_j = \text{sign}((\mathbf{w})_{j,k-1})((\mathbf{w})_{j,k-1})^2/(\mathbf{u})_{j,k-1}$ or $(\mathbf{r})_j = 0$ if $(\mathbf{u})_{j,k-1} = 0$
5      Compute the $p \times 1$ vector $\mathbf{J}$. The elements of $\mathbf{J}$ are the ordered indices of the
         predictors (ordered first according to group, and then within group according to
         the elements of $\mathbf{r}$.)
6      Initialize $R_{best} = 0$
7      for $j = 1, \ldots, p$ do
8          If $(\mathbf{J})_j$ corresponds to a new group, reset $\mathbf{z}_{old} = \mathbf{z}_{new} = \mathbf{0}_{n\times 1}$,
             $N_{old} = D_{old} = N_{new} = D_{new} = 0$
9          $(\mathbf{q})_{(\mathbf{J})_j} = \mathbf{e}_{(\mathbf{J})_j}^{\mathsf{T}}\mathbf{z}_{old}$
10         $\mathbf{z}_{new} = \mathbf{z}_{old} + \mathbf{x}_{(\mathbf{J})_j}$
11         $N_{new} = N_{old} + (\mathbf{w})_{(\mathbf{J})_j}$ and $D_{new} = D_{old} + 2(\mathbf{q})_{(\mathbf{J})_j} + (\mathbf{u})_{(\mathbf{J})_j}$
12         $R_{new} = N_{new}^2/D_{new}$ (i.e., Equation (12))
13         Update $D_{old} = D_{new}$, $N_{old} = N_{new}$ and $\mathbf{z}_{old} = \mathbf{z}_{new}$
14         If $R_{new} > R_{best}$, set $R_{best} = R_{new}$ and $\mathbf{z}_{best} = \mathbf{z}_{new}$
15     $\mathbf{z}_k = \mathbf{z}_{best}$
16     $\mathbf{e}_{\mathbf{z}_k,k-1} = \mathbf{z}_k - \mathbf{Z}_{k-1}[\mathbf{Z}_{k-1}^{\mathsf{T}}\mathbf{Z}_{k-1}]^{-1}\mathbf{Z}_{k-1}^{\mathsf{T}}\mathbf{z}_k$
17     $\mathbf{v}_k = \mathbf{E}_{k-1}^{\mathsf{T}}\mathbf{e}_{\mathbf{z}_k,k-1}$
18     $\mathbf{E}_k = \mathbf{E}_{k-1} - \frac{\mathbf{e}_{\mathbf{z}_k,k-1}}{\mathbf{e}_{\mathbf{z}_k,k-1}^{\mathsf{T}}\mathbf{e}_{\mathbf{z}_k,k-1}}{\mathbf{v}_k}^{\mathsf{T}}$ (i.e., Equation (16))
19     $\mathbf{w}_k = \mathbf{w}_{k-1} - \mathbf{v}_k\frac{\mathbf{e}_{\mathbf{z}_k,k-1}^{\mathsf{T}}\mathbf{y}}{\mathbf{e}_{\mathbf{z}_k,k-1}^{\mathsf{T}}\mathbf{e}_{\mathbf{z}_k,k-1}}$ and $\mathbf{u}_k = \mathbf{u}_{k-1} - \frac{\mathbf{v}_k \circ \mathbf{v}_k}{\mathbf{e}_{\mathbf{z}_k,k-1}^{\mathsf{T}}\mathbf{e}_{\mathbf{z}_k,k-1}}$ (i.e.,
         Equation (17))
```

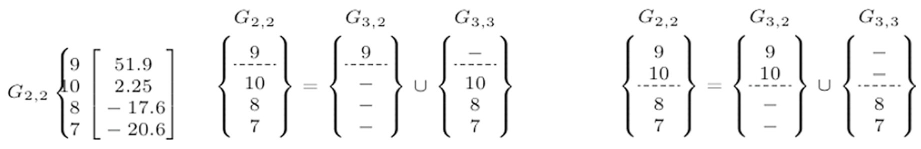
As shown in Sect. 2.3, splitting a group of predictors at iteration k is equivalent to adding a new derived feature $\mathbf{z}$ to the basis, and the reduction $R(\mathbf{z})$ in SSE can be efficiently computed for any potential split. In this section, we discuss a greedy heuristic algorithm for finding the $\mathbf{z}$ that approximately maximizes $R(\mathbf{z})$ when deciding how to best split any group of predictors (say $G_{k-1,i}$) at iteration k. Algorithm 1 gives the pseudo-code for the CTR procedure.

At each iteration k, there are k groups (including the group $G_{k-1,k}$ of omitted predictors) to search for possible splitting, and p possible split points across all k groups. For each $i = 1, 2, \dots, k$, the split-point search procedure passes over a sorted version of the predictors in group $G_{k-1,i}$. The sorting criterion is based on the partial correlations between the p predictors and the response, after regressing out their dependencies on the derived features $\mathbf{Z}_{k-1}$, and is explained in more detail later in this section. After ordering the predictors in $G_{k-1,i}$, the split-point search procedure sequentially scans through the ordered elements of $G_{k-1,i}$ for the optimal split point. In Algorithm 1, we use the vector $\mathbf{z}_{best}$ to denote the new derived feature obtained at the end of iteration k, and the vectors $\mathbf{z}_{new}$ and $\mathbf{z}_{old}$ to represent the derived features at the current and previous steps of the split-point search, respectively. Here, each "step" corresponds to moving the split point down by one predictor j included in the sorted $G_{k-1,i}$, so that $\mathbf{z}_{new}$ is obtained from $\mathbf{z}_{old}$ by just adding the single predictor j (i.e., $\mathbf{z}_{new} = \mathbf{z}_{old} + \mathbf{x}_j$). Returning to the example of Fig. 1 to illustrate the split-point search procedure, during iteration $k = 3$, we seek a group to split by considering each of the existing groups $G_{2,1}$, $G_{2,2}$ and $G_{2,3}$ obtained after iteration $k - 1 = 2$. It turned out that $G_{2,i} = G_{2,2}$ is the group whose split maximized $R(\mathbf{z})$, so we illustrate the split-point search procedure with group $G_{2,2} = \{7, 8, 9, 10\}$ in Fig. 3.

As mentioned above, the split-point search procedure updates the derived feature sequentially, at each step adding the next predictor in the ordered sequence of p predictors. We sort the predictors within group $G_{k-1,i}$ in either ascending or descending order based on a rule depending on the marginal reduction in SSE for predictor $\mathbf{x}_j$ for $j \in G_{k-1,i}$, i.e., the reduction in SSE if $\mathbf{z}_{old}$ were a vector of zeros and $\mathbf{z}_{new} = \mathbf{x}_j$. Thus, based on our sorting criterion, the predictors having larger marginal reduction in SSE (and marginal effects on the response that are of the same sign) are good candidates to be included in the same group. The rationale behind the sorting criterion is described more in Sect. 2.5.

Returning to our example at iteration $k = 3$, when searching over the group $G_{2,2} = \{7, 8, 9, 10\}$ to split into two subgroups $G_{3,2}$ and $G_{3,3}$, the aim is to find the derived feature $\mathbf{z}_{3,2}$ (the candidates for which are denoted $\mathbf{z}_{new}$ in Algorithm 1) that most reduces the SSE, with $\mathbf{z}_{2,2} = \mathbf{z}_{3,2} + \mathbf{z}_{3,3}$. We first order the predictors in $G_{2,2}$ according to our sorting criterion, and obtain the ordered $G_{2,2} = \{9, 10, 8, 7\}$. Figure 3b, c display the ordered group $G_{2,2}$ and the groups $G_{3,2}$ and $G_{3,3}$ constructed after the first two steps. The procedure then sequentially scans through all three potential split points in the ordered $G_{2,2}$ (two of which are shown in Fig. 3; the details about the ordering illustrated in Fig. 3a are explained in the next section).

For the general situation, the procedure chooses the best split point in the ordered sequence of predictors as the one that results in the largest $R(\mathbf{z}_{new})$. Recall that we search every existing group for possible splitting. The number of possible split points within the group $G_{k-1,i}$ is its cardinality $|G_{k-1,i}|$, if we consider the first split point to correspond to no split at all (for which $R(\mathbf{z}_{new}) = 0$). This is implemented in lines 7–14 of Algorithm 1, in which $\mathbf{z}_{best}$ stores the $\mathbf{z}_{new}$ vector for which $R(\mathbf{z}_{new})$ was the highest out of all possible split points that have been tried. In Sect. 2.5, the procedure to efficiently update $R(\mathbf{z}_{new})$ from $R(\mathbf{z}_{old})$ is described.



(a) $(\mathbf{r})_{7:10}(\times 10^2)$ **(b)** Split after the first step ($\mathbf{z}_{3,2} = \mathbf{x}_9$ and $\mathbf{z}_{3,3} = \mathbf{x}_{10} + \mathbf{x}_8 + \mathbf{x}_7$) **(c)** Split after the second step ($\mathbf{z}_{3,2} = \mathbf{x}_9 + \mathbf{x}_{10}$ and $\mathbf{z}_{3,3} = \mathbf{x}_8 + \mathbf{x}_7$)

Fig. 3 For the Fig. 1 example, illustration of the ordering of the predictors and split-point search at iteration $k = 3$ for finding $\mathbf{z}_{best} = \mathbf{z}_{3,2}$ by splitting the group $G_{2,2} = \{7, 8, 9, 10\}$. The numbers to the left of the last four elements of vector $\mathbf{r}$ in Fig. 3a are the predictor indices.

In our example shown in Fig. 3b (prior to which the first group $G_{2,1}$ had already been searched for its best split point), the split-point search over $G_{2,2}$ begins by resetting $\mathbf{z}_{old}$ and $\mathbf{z}_{new}$ to zero vectors and then checking the first ordered predictor $\mathbf{x}_9$ in $G_{2,2}$. The latter is done by setting $\mathbf{z}_{new} = \mathbf{x}_9$ (i.e., $G_{3,2} = \{9\}$). Since $R(\mathbf{z}_{new}) = 5193.5 > R(\mathbf{z}_{best})$ ($R(\mathbf{z}_{best}) = 22.5$ after the best split of group $G_{2,1}$), the procedure updates $\mathbf{z}_{best} = \mathbf{z}_{new}$. The two subgroups $G_{3,2} = \{9\}$ and $G_{3,3} = \{10, 8, 7\}$ are therefore formed after this step. The procedure next checks the predictor $\mathbf{x}_{10}$, as it is the next predictor in the ordered $G_{2,2}$ group, by setting $\mathbf{z}_{new} = \mathbf{x}_9 + \mathbf{x}_{10}$ (i.e., $G_{3,2} = \{9, 10\}$) as shown in Fig. 3c. Since $R(\mathbf{z}_{new}) = 5659.8 > R(\mathbf{z}_{best}) = 5193.5$, the procedure updates $\mathbf{z}_{best} = \mathbf{z}_{new}$. The subgroups are now $G_{3,2} = \{9, 10\}$ and $G_{3,3} = \{8, 7\}$. Next, the procedure checks $\mathbf{x}_8$ and $\mathbf{x}_7$, sequentially, but $R(\mathbf{z}_{new}) < R(\mathbf{z}_{best})$ for both of these last two split points. Thus, $\mathbf{z}_{best} = \mathbf{z}_{3,2} = \mathbf{x}_9 + \mathbf{x}_{10}$ represents the best split of $G_{2,2}$, and the two subgroups are $G_{3,2} = \{9, 10\}$ and $G_{3,3} = \{8, 7\}$. The procedure then resets $\mathbf{z}_{old}$ and $\mathbf{z}_{new}$ to zero vectors and repeats the sequential search to find the best split of $G_{2,3}$. For this example, the group whose best split reduced the SSE the most is $G_{2,2}$, and its split into $G_{3,2} = \{9, 10\}$ and $G_{3,3} = \{8, 7\}$ was taken to be the best overall split at iteration $k = 3$.

2.5 Updating the model efficiently at the end of each iteration

At each step of the split-point search, one can compute the reduction $R(\mathbf{z}_{new})$ in SSE going from derived features $\mathbf{Z}_{k-1}$ to $[\mathbf{Z}_{k-1}, \mathbf{z}_{new}]$ via (8), and thus it becomes

$$R(\mathbf{z}_{new}) = \frac{(\mathbf{e}^{\top}_{\mathbf{z}_{new},k-1}\mathbf{e}_{\mathbf{y},k-1})^2}{\mathbf{e}^{\top}_{\mathbf{z}_{new},k-1}\mathbf{e}_{\mathbf{z}_{new},k-1}} \equiv \frac{N^2_{new}}{D_{new}}. \tag{9}$$

However, we take the advantage of sequential update during the split-point search procedure (i.e., recall that the new derived feature $\mathbf{z}_{new}$ is updated sequentially at each step by adding a single predictor onto the existing derived feature $\mathbf{z}_{old}$) to reduce the computational expense compared the existing methods. In this section, we explain how to update $R(\mathbf{z}_{new})$ efficiently during the split-point search procedure and the model at the end of each iteration.

During the split-point search procedure, we can efficiently update $R(\mathbf{z}_{new})$ as follows. Let $\mathbf{x}_j$ denote the new predictor that is added at the current split-point search step, so that $\mathbf{z}_{new} = \mathbf{z}_{old} + \mathbf{x}_j$. For notational simplicity, define $\mathbf{e}_{j,k-1} = \mathbf{e}_{\mathbf{x}_j,k-1}$. Suppose that we have already calculated the numerator and denominator of $R(\mathbf{z}_{old}) = \frac{N^2_{old}}{D_{old}}$ in (8), and we want to calculate $R(\mathbf{z}_{new})$. Then, we have

$$\mathbf{e}_{\mathbf{z}_{new},k-1} = \left[\mathbf{I} - \mathbf{P}_{k-1}\right]\mathbf{z}_{new} = \left[\mathbf{I} - \mathbf{P}_{k-1}\right](\mathbf{z}_{old} + \mathbf{x}_j) = \mathbf{e}_{\mathbf{z}_{old},k-1} + \mathbf{e}_{j,k-1}, \tag{10}$$

and plugging (10) into (9) gives

$$R(\mathbf{z}_{new}) = \frac{(N_{old} + \mathbf{e}_{j,k-1}^{\mathsf{T}}\mathbf{e}_{\mathbf{y},k-1})^2}{D_{old} + 2\mathbf{e}_{j,k-1}^{\mathsf{T}}\mathbf{e}_{\mathbf{z}_{old},k-1} + \mathbf{e}_{j,k-1}^{\mathsf{T}}\mathbf{e}_{j,k-1}}, \tag{11}$$

where N_{old} and D_{old} are available from the previous step of the split-point search. Since $\mathbf{e}_{j,k-1}^{\mathsf{T}}\mathbf{e}_{\mathbf{y},k-1} = \mathbf{e}_{j,k-1}^{\mathsf{T}}\mathbf{y}$ and $\mathbf{e}_{j,k-1}^{\mathsf{T}}\mathbf{e}_{\mathbf{z}_{old},k-1} = \mathbf{e}_{j,k-1}^{\mathsf{T}}\mathbf{z}_{old}$, we can also write (11) as

$$R(\mathbf{z}_{new}) = \frac{(N_{old} + \mathbf{e}_{j,k-1}^{\mathsf{T}}\mathbf{y})^2}{D_{old} + 2\mathbf{e}_{j,k-1}^{\mathsf{T}}\mathbf{z}_{old} + \mathbf{e}_{j,k-1}^{\mathsf{T}}\mathbf{e}_{j,k-1}}. \tag{12}$$

As mentioned above, we search each existing group for possible splitting during iteration k, and the split-point search passes over a sorted version of the predictors in each group in $\{G_{k-1,i} : i = 1, 2, \dots, k\}$. Regarding the ordering, the rationale is that, according to (12), predictors with the largest values for $\frac{(\mathbf{e}_{j,k-1}^{\mathsf{T}}\mathbf{y})^2}{\mathbf{e}_{j,k-1}^{\mathsf{T}}\mathbf{e}_{j,k-1}}$ are good candidates for reducing the SSE when included in the derived feature $\mathbf{z}_{new}$, and if they are to be in the same group, they should have the same sign $\mathbf{e}_{j,k-1}^{\mathsf{T}}\mathbf{y}$. Therefore, we order the predictors within group $G_{k-1,i}$ in either ascending or descending order of the corresponding elements of the $p \times 1$ vector $\mathbf{r}$, the jth element of which is

$$(\mathbf{r})_j = \frac{\text{sign}(\mathbf{e}_{j,k-1}^{\mathsf{T}}\mathbf{y})(\mathbf{e}_{j,k-1}^{\mathsf{T}}\mathbf{y})^2}{\mathbf{e}_{j,k-1}^{\mathsf{T}}\mathbf{e}_{j,k-1}} \tag{13}$$

(ascending if the largest squared "partial correlation" $\frac{(\mathbf{e}_{j,k-1}^{\mathsf{T}}\mathbf{y})^2}{\mathbf{e}_{j,k-1}^{\mathsf{T}}\mathbf{e}_{j,k-1}}$ within the group has negative $\mathbf{e}_{j,k-1}^{\mathsf{T}}\mathbf{y}$, and descending otherwise; see line 5 in Algorithm 1). Note that $\frac{(\mathbf{e}_{j,k-1}^{\mathsf{T}}\mathbf{y})^2}{\mathbf{e}_{j,k-1}^{\mathsf{T}}\mathbf{e}_{j,k-1}}$ is the marginal reduction in SSE for $\mathbf{x}_j$, i.e., the reduction in SSE if $\mathbf{z}_{old}$ were a vector of zeros and $\mathbf{z}_{new} = \mathbf{x}_j$, and thus the predictors having larger marginal reduction in SSE in magnitude are good candidates to be included in the same group as mentioned in Sect. 2.4.

Returning to our example illustrated in Fig. 3 at iteration $k = 3$, when searching over the group $G_{2,2} = \{7, 8, 9, 10\}$ to split into two subgroups $G_{3,2}$ and $G_{3,3}$, we first order the predictors in $G_{2,2}$ according to their corresponding elements of $\mathbf{r}$, which are $(\mathbf{r})_{7:10} = 10^2 \times \left[-20.6, -17.6, 51.9, 2.25\right]$. The largest magnitude element is +51.9, so we order the last four elements of $\mathbf{r}$ in descending order as illustrated in Fig. 3a.

For computational purposes, after finding the new derived feature $\mathbf{z}_k$ at the end of kth iteration, we update and store the $n \times p$ matrix $\mathbf{E}_k = [\mathbf{e}_{1,k}, \mathbf{e}_{2,k}, \dots, \mathbf{e}_{p,k}]$ and the $p \times 1$ vectors $\mathbf{w}_k$ and $\mathbf{u}_k$, the jth elements of which are

$$(\mathbf{w})_{j,k} = \mathbf{e}_{j,k}^{\mathsf{T}}\mathbf{y} \text{ and } (\mathbf{u})_{j,k} = \mathbf{e}_{j,k}^{\mathsf{T}}\mathbf{e}_{j,k}. \tag{14}$$

These are to be used in the next iteration ($k \to k + 1$) of Algorithm 1. Since the columns of $\mathbf{E}_k$ and $\mathbf{E}_{k-1}$ are related via

$$
\begin{aligned}
\mathbf{e}_{j,k} = \mathbf{x}_j - \mathbf{P}_k\mathbf{x}_j = \mathbf{x}_j - \mathbf{P}_{k-1}\mathbf{x}_j - \frac{\mathbf{e}^{\top}_{\mathbf{z}_k,k-1}\mathbf{x}_j}{\mathbf{e}^{\top}_{\mathbf{z}_k,k-1}\mathbf{e}_{\mathbf{z}_k,k-1}}\mathbf{e}_{\mathbf{z}_k,k-1} \\
= \mathbf{e}_{j,k-1} - \frac{\mathbf{e}^{\top}_{\mathbf{z}_k,k-1}\mathbf{e}_{j,k-1}}{\mathbf{e}^{\top}_{\mathbf{z}_k,k-1}\mathbf{e}_{\mathbf{z}_k,k-1}}\mathbf{e}_{\mathbf{z}_k,k-1},
\end{aligned}
\tag{15}
$$

at the end of the kth iteration the $n \times p$ matrix $\mathbf{E}_k$ can be updated from $\mathbf{E}_{k-1}$ via

$$
\mathbf{E}_k = \mathbf{E}_{k-1} - \frac{\mathbf{e}_{\mathbf{z}_k,k-1}}{\mathbf{e}^{\top}_{\mathbf{z}_k,k-1}\mathbf{e}_{\mathbf{z}_k,k-1}}\mathbf{v}_k^{\top},
\tag{16}
$$

where the $p \times 1$ vector $\mathbf{v}_k = \mathbf{E}^{\top}_{k-1}\mathbf{e}_{\mathbf{z}_k,k-1}$. After calculating $\mathbf{v}_k$, the vectors $\mathbf{w}_k$ and $\mathbf{u}_k$ can be updated via

$$
\begin{aligned}
\mathbf{w}_k = \mathbf{w}_{k-1} - \mathbf{v}_k\frac{\mathbf{e}^{\top}_{\mathbf{z}_k,k-1}\mathbf{y}}{\mathbf{e}^{\top}_{\mathbf{z}_k,k-1}\mathbf{e}_{\mathbf{z}_k,k-1}} \text{ and} \\
\mathbf{u}_k = \mathbf{u}_{k-1} - \frac{\mathbf{v}_k \circ \mathbf{v}_k}{\mathbf{e}^{\top}_{\mathbf{z}_k,k-1}\mathbf{e}_{\mathbf{z}_k,k-1}},
\end{aligned}
\tag{17}
$$

which follows from their definitions in (14) and from (15) and (16). Here, $\circ$ is the element-wise product operation. The preceding provides an efficient means of updating $\mathbf{E}_k$, $\mathbf{w}_k$ and $\mathbf{u}_k$ at the end of iteration k.

2.6 Choosing the termination criterion

The CTR algorithm sequentially adds derived features that are the sums of predictors within the discovered groups. After each iteration k, the number of derived features in the model is k, and there are k levels in the corresponding graphical tree output (as in Fig. 1). Consequently, the CTR algorithm requires a termination criterion that represents how large the tree should be grown (equivalently, the number of derived features in the model at termination). The tree size is the only tuning parameter for CTR, and this can be chosen using K-fold CV as described in the remainder of this section.

In all of our experiments, we use 10-fold CV to select the best k. We first split the data into 10 equal-size folds. For the ith fold, we apply the CTR algorithm to the other nine folds and grow the tree up to some sufficiently large model size k_{max} that represents a conservative upper bound on the best model size. For each model in this sequence of fitted CTR trees with k ranging from one to k_{max}, we calculate the CV SSE for predicting the ith hold-out fold. We repeat this for each hold-out fold ($i = 1, \dots, 10$) and then sum the CV hold-out SSEs to give a CV SSE for each of the k_{max} models (i.e., with size ranging from $k = 1, 2, \dots, k_{max}$). We then choose the best k as the one that minimizes the CV SSE. After choosing the best k in this manner, we fit the final CTR model of this chosen size k using all the data. Returning to the toy example in Fig. 1, we compute the 10-fold CV SSE for tree size ranging over $k = 1, 2, \dots, 10$ as illustrated in Fig. 2, which is minimized at $k = 4$.

The user must select k_{max}, and for this we suggest the following considerations. We used $k_{max} = 20$ in all of our examples. A smaller model size is usually sufficient even when there are many more true groups (or in the more extreme case, when there is no group structure at all, so that the number of groups is p) for the reasons that we discuss in Sect. 5 and illustrate in Fig. 19. In addition to the experiments presented in this paper, the studies in the prior literature have shown that oftentimes a simpler, parsimonious model is enough to accurately predict the response variable (Hand, 2006; Verbeke et al., 2012; Zhao et al., 2014). However, in some cases a better model might be obtained with larger model size (e.g., $k > 20$). This can be easily determined as follows. If we begin with some initial k_{max} (say, $k_{max} = 20$, which would be viewed as a preliminary upper bound for k), and the best k according to the above CV procedure is the maximum value k_{max}, this is an indication that $k > k_{max}$ may be better. In this case, one should increase k_{max} to a larger value (i.e., $k_{max} = 40$) and compare the CV SSE of the larger fitted models. After increasing k_{max} in this manner, instead of running the CTR algorithm and CV again from scratch, one can use a "warm-start" strategy in which the CTR output for the initial k_{max} (i.e., the fitted models of size $k = 1, 2, \ldots, k_{max}$, the CV SSE values for each of these models, the CV partition indices, etc.) are given as inputs to the continuation of the CTR algorithm. We have implemented this warm-start feature in our `CTR` R package, which reduces the time required to train a model.

3 Predictive performance and computational expense comparisons

3.1 Predictive performance comparison on synthetic data

In this section, we investigate the accuracy of CTR on simulated examples under various conditions. The performance of the CTR algorithm is compared with the other group methods (fused lasso and OSCAR). We also include regular lasso and OLS to serve as references for comparison, even though they are not group methods and do not create features from the predictors. We used the publicly available `R` language package `glmnet` (Friedman et al., 2010) for lasso, `lm` for OLS (Team, 2017), and `lqa` package (Ulbricht, 2012) for OSCAR and fused lasso. For fused lasso, since a preliminary ordering is required, we order the predictors based on the least squares estimates. We omitted CARDS from the performance comparisons because we were unable to obtain any software package to implement it.

To implement CTR, we used our own `CTR` R package. The CTR R package is available from https://ozgesurer.github.io/files/ctr-vignette.html. The vignette (https://ozgesurer.github.io/OS/ctr-vignette.html) illustrates how the package can be installed and used via a small example. The hardware consisted of a computer with macOS Sierra operating system and 2.5 GHz processor with 16 GB RAM. The model complexity parameters for lasso (a regularization parameter), OSCAR and fused lasso (two regularization parameters), and CTR (the number k of derived features) were all chosen using CV. In order to find the best k for CTR, we obtained the CV SSE with model size ranging from 1 to k_{max} similar to how it is obtained with the toy example shown in Fig. 2, and then the one with minimum CV SSE is chosen as the best model size. Our simulations compared the accuracy of different models on data generated from the regression model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathrm{N}(\mathbf{0}, \sigma^2\mathbf{I})$ and each row of $\mathbf{X}$ is $\mathrm{MVN}_p(\mathbf{0}, \boldsymbol{\Sigma})$. We generated different experiments based on true $\boldsymbol{\beta}$ explained later in detail. We used $p = 1000$ predictor variables for the accuracy comparisons and larger p for some of the run-time comparisons described in Sect. 3.2. We varied the number of observations as $n \in \{1500, 2000, 3000\}$ and represented varying noise levels by using a true $r^2 \in \{0.5, 0.7, 0.9\}$, where true $r^2 = \frac{\mathbb{V}(\mathbf{y}-\boldsymbol{\epsilon})}{\mathbb{V}(\mathbf{y})}$. For the accuracy comparisons, for each experiment (i.e., for each combination of true model, n, and true r^2) we generated one test set of 10,000 observations and 100 training data sets of size n, the latter representing 100 replicates of the experiment. On each replicate we fitted CTR, lasso, fused lasso, OSCAR and OLS models to the training data and applied the estimated models to predict the test set. We summarized the accuracy of the models over the 100 replicates by averaging the test r^2 values (denoted $\bar{r}^2$).

The experiments fall into three categories based on the true $\boldsymbol{\beta}$: A true group structure without sparsity, a true group structure with sparsity, and a true non-group structure. The latter was included to demonstrate that CTR still performs quite well, and much better than OLS, even when there is no actual group structure in the predictors (i.e., when there are no nonzero-coefficient predictors sharing the same coefficient). We consider similar settings as the simulation studies presented in (Bondell and Reich, 2008; Shen and Huang, 2010; Zhu et al.,2013; Ke et al., 2015). For each category, we consider different $\boldsymbol{\beta}$ and $\boldsymbol{\Sigma}$ structures, the test $\bar{r}^2$ values for which are plotted in Figs. 4, 5, 6 and 7. Columns of these figures correspond to different true r^2 values, and different values of n are shown within each panel. Each row of panels in each figure are for various different values of $\boldsymbol{\beta}$ and $\boldsymbol{\Sigma}$ as follows. First, we consider a model with true group structure and no sparsity and with true coefficients $\boldsymbol{\beta}^\top = \left[\mathbf{1}^\top_{p/2}, -\mathbf{1}^\top_{p/2}\right]$, $\boldsymbol{\beta}^\top = \left[\mathbf{10}^\top_{10}, \mathbf{1}^\top_{p-10}\right]$, and $\boldsymbol{\beta}^\top = \left[\mathbf{5}^\top_{p/5}, \mathbf{4}^\top_{p/5}, \mathbf{3}^\top_{p/5}, \mathbf{2}^\top_{p/5}, \mathbf{1}^\top_{p/5}\right]$, each with $\boldsymbol{\Sigma} = \mathbf{I}$ (Fig. 4). True models with group structure and sparsity are then examined with true coefficients $\boldsymbol{\beta}^\top = \left[\mathbf{1}^\top_{50}, \mathbf{0}^\top_{p-50}\right]$, $\boldsymbol{\beta}^\top = \left[\mathbf{10}^\top_{5}, \mathbf{1}^\top_{p/2}, \mathbf{0}^\top_{p-p/2-5}\right]$, and $\boldsymbol{\beta}^\top = \left[\mathbf{1}^\top_{0.3p}, \mathbf{0}^\top_{0.4p}, -\mathbf{1}^\top_{0.3p}\right]$ with $\boldsymbol{\Sigma} = \mathbf{I}$ (Fig. 5). Finally, true models with no group structure are investigated with $\beta_j = (j - 0.5p)/100$ for $j = 1, \dots, p$ and $\boldsymbol{\Sigma} = \mathbf{I}$, $\beta_j = (j - 0.5p)/100$ for $j = 1, \dots, p$ and $\boldsymbol{\Sigma}_{i,j} = 0.7^{|i-j|}$, $\beta_j = (j - 0.5p)/100$ for $j = 1, \dots, p/2$, $\beta_j = 0$ for $j = p/2 + 1, \dots, p$ and $\boldsymbol{\Sigma} = \mathbf{I}$ (Fig. 6), and $\beta_j = (j - p)/100$ for $j = 1, \dots, p/2$, $\beta_j = j/100$ for $j = p/2 + 1, \dots, p$ and $\boldsymbol{\Sigma} = \mathbf{I}$, $\beta_j = (j - p)/100$ for $j = 1, \dots, p/2$, $\beta_j = j/100$ for $j = p/2 + 1, \dots, p$ and $\boldsymbol{\Sigma}_{i,j} = 0.7^{|i-j|}$, $\beta_j = (j - p)/100$ for $j = 1, \dots, p/4$, $\beta_j = 0$ for $j = p/4 + 1, \dots, 3p/4$, $\beta_j = j/100$ for $j = 3p/4 + 1, \dots, p$ and $\boldsymbol{\Sigma} = \mathbf{I}$ (Fig. 7). Figure 6 shows the performance when the coefficients in $\boldsymbol{\beta}$ vary continuously (i.e., no true group structure), and Fig. 7 illustrates the performance when the true model has no true group structure and the coefficients do not change smoothly.

The following are the main takeaways from the simulation experiment results. In terms of *predictive performance*, CTR was almost always substantially better than the other group-based methods (OSCAR and fused lasso) and also than OLS, which demonstrates the power of CTR at finding relevant features that are groups of predictors. The overall predictive performance of CTR across all experiments was even comparable to, or a little better than, that of the non-group-based lasso benchmark, with each method sometimes coming out on top. Occasionally, especially for smaller true r^2 and smaller n, the test $\bar{r}^2$ for OSCAR, fused lasso, and OLS was negative, which indicates severe overfitting. None of the CTR or lasso models ever had negative $\bar{r}^2$, which is an indication that CV effectively selected their complexity parameters.

Comparing the predictive performance in more detail, for the Fig. 4 experiments (true group structure without sparsity), CTR was almost always better than lasso, sometimes by a little and sometimes by a lot. For the Fig. 5 experiments (true group structure with sparsity), CTR and lasso performed comparably, with perhaps a slight advantage overall for CTR. Even for the experiments in Fig. 6, for which there is no true group structure, CTR performed comparably to lasso. For the top row of experiments, CTR and lasso were comparable. While for the middle row of experiments, CTR is a little better than lasso, and for the bottom row of experiments, CTR is a little worse than lasso, presumably because there

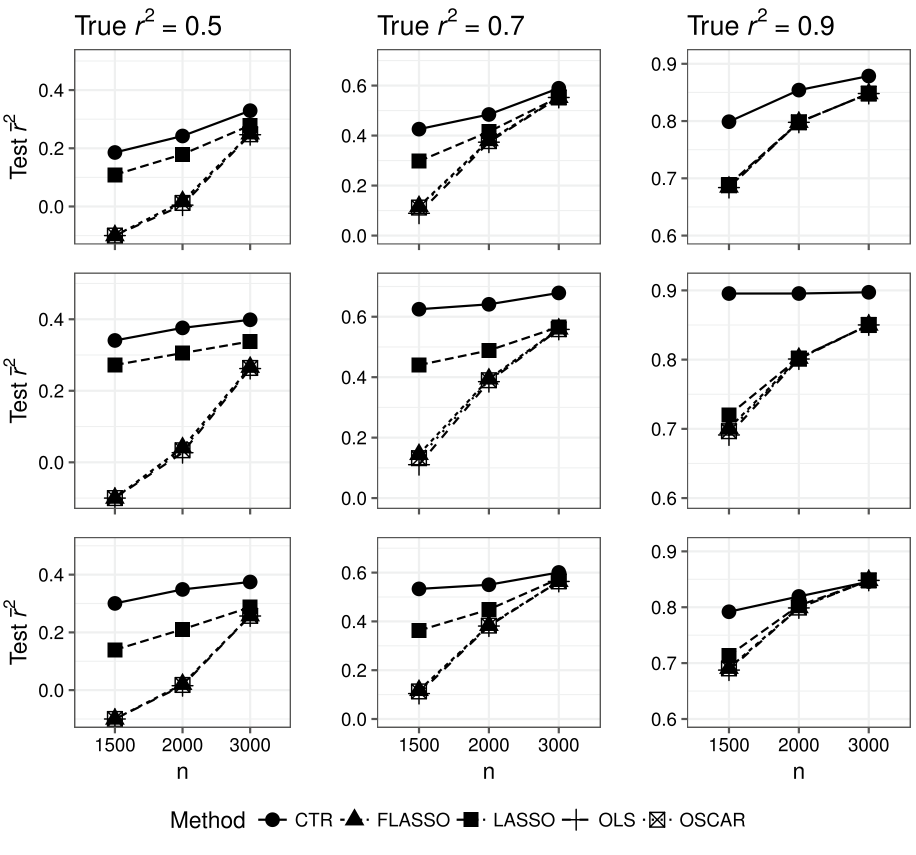


Fig. 4 Simulation results with true group structure and no sparsity with $\boldsymbol{\beta}^{\top} = [\mathbf{1}^{\top}_{p/2}, -\mathbf{1}^{\top}_{p/2}]$ (1st row), $\boldsymbol{\beta}^{\top} = [\mathbf{10}^{\top}_{10}, \mathbf{1}^{\top}_{p-10}]$ (2nd row), and $\boldsymbol{\beta}^{\top} = [\mathbf{5}^{\top}_{p/5}, \mathbf{4}^{\top}_{p/5}, \mathbf{3}^{\top}_{p/5}, \mathbf{2}^{\top}_{p/5}, \mathbf{1}^{\top}_{p/5}]$ (3rd row), all with $\boldsymbol{\Sigma} = \mathbf{I}$. Note that LASSO is not a group method and is only included as a reference for comparison

was true sparsity in the bottom row example. In all of our experiments CTR is much better than OLS, OSCAR and fused lasso, especially for smaller true r^2 and smaller n. In terms of *interpretability*, CTR shares lasso's sparsity advantages when there truly is sparsity, as in the Fig. 5 experiments, but also has the additional important advantage of group sparsity when there is a true group structure. For example, in the top row of experiments in Fig. 5 ($\boldsymbol{\beta}^{\top} = [\mathbf{1}^{\top}_{50}, \mathbf{0}^{\top}_{p-50}]$), CTR only needs a single derived feature, whereas lasso still requires all 50 predictors. And in the experiments in Fig. 4, for which there is no sparsity, CTR only requires a few derived features, whereas lasso requires all $p = 1000$ predictors. In each case, CTR's derived features are simple sums of groups of predictors, which are often easy to interpret.

In Figs. 4, 5, 6 and 7, the predictive performance is summarized for $n \in \{1500, 2000, 3000\}$. We next compare the performance of CTR and lasso for larger and smaller n values as follows. Because the predictive performance of both CTR and lasso is better than OSCAR and fused lasso in all the experiments when $n = 1500$ (due to overfitting, the latter two often have negative $\bar{r}^2$ when the true r^2 is low) and the computational time increases significantly for both OSCAR and fused lasso when n gets larger, we

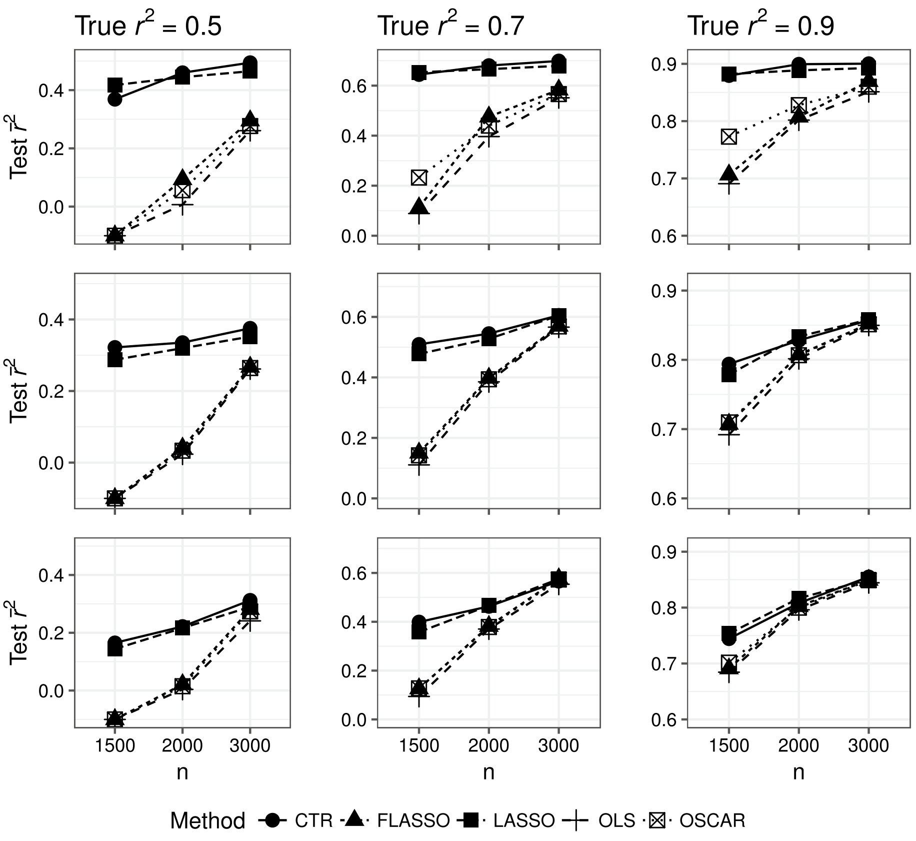


Fig. 5 Simulation results with true group structure and sparsity with $\boldsymbol{\beta}^\top = [\mathbf{1}_{50}^\top, \mathbf{0}_{p-50}^\top]$ (1st row), $\boldsymbol{\beta}^\top = [\mathbf{10}_5^\top, \mathbf{1}_{p/2}^\top, \mathbf{0}_{p-p/2-5}^\top]$ (2nd row), and $\boldsymbol{\beta}^\top = [\mathbf{1}_{0.3p}^\top, \mathbf{0}_{0.4p}^\top, -\mathbf{1}_{0.3p}^\top]$ (3rd row), all with $\boldsymbol{\Sigma} = \mathbf{I}$. Note that LASSO is not a group method and is only included as a reference for comparison

only provide the results for two competitive methods, CTR and lasso. Aside from having different n, we consider the experiments in Figs. 4, 5 and 6, and the results are displayed in Table 1.

3.2 Computational expense comparison

We considered larger data sets when investigating the computational performance of CTR. For these experiments, we used the setting in the middle row of experiments in Fig. 6 (no true group structure and correlated predictors) with $n \in \{10^3, 10^4, 10^5, 10^6\}$ and $p \in \{500, 1000, 2000, 4000\}$. Table 2 reports the average computation times (in seconds) across ten replicates of each experiment (there was low replicate-to-replicate variability). The computation times for CTR, lasso, OSCAR and fused lasso include 10-fold CV to select their model complexity parameter. For fused lasso and OSCAR, we provided the set of values $\{0.005, 0.05, 0.5\}$ for each penalty parameter.

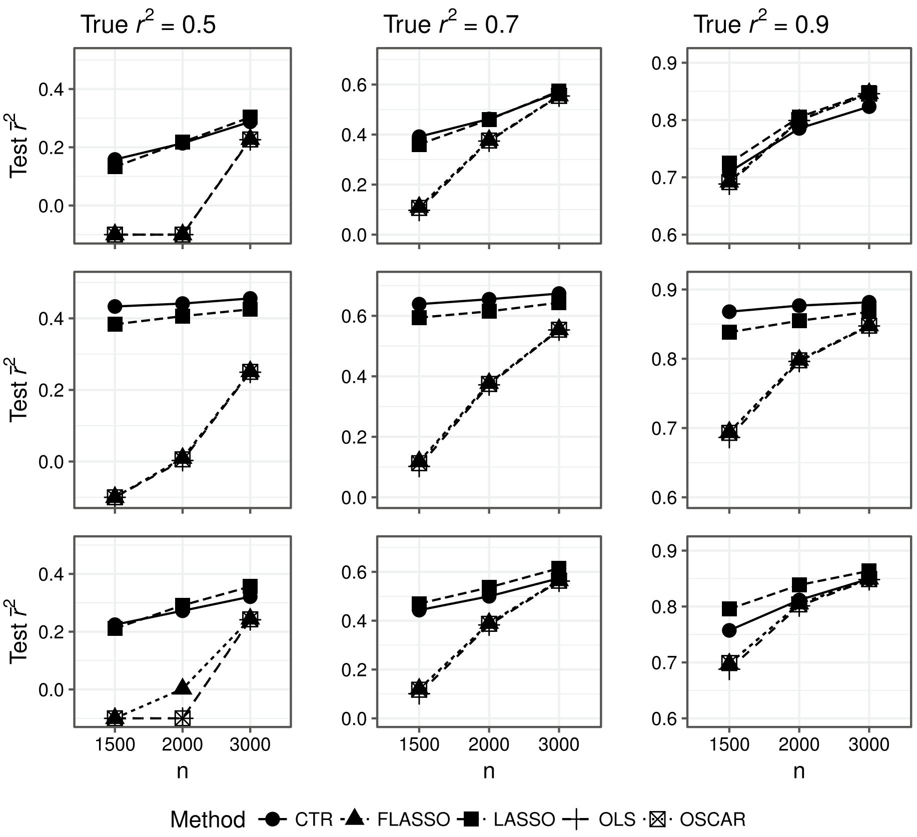


Fig. 6 Simulation results with no true group structure with $\beta_j = (j - 0.5p)/100$ for $j = 1, \ldots, p$ and $\mathbf{\Sigma} = \mathbf{I}$ (1st row), $\beta_j = (j - 0.5p)/100$ for $j = 1, \ldots, p$ and $\mathbf{\Sigma}_{i,j} = 0.7^{|i-j|}$ (2nd row), $\beta_j = (j - 0.5p)/100$ for $j = 1, \ldots, p/2$, $\beta_j = 0$ for $j = p/2 + 1, \ldots, p$ and $\mathbf{\Sigma} = \mathbf{I}$ (3rd row). Note that LASSO is not a group method and is only included as a reference for comparison

In terms of *computational expense*, CTR and the `glmnet` implementation (which is considered state-of-the-art and extremely fast) of lasso are all $O(np)$ theoretically (see Sect. 5), in contrast to a brute-force implementation ($O(np^2) + O(p^3)$) or QR-based implementation ($O(np^2)$) of OLS. However, CTR is almost an order-of-magnitude faster than lasso for the examples in Table 2. It should be noted that, in order to have a common basis for comparison, the computational results in Table 2 are without using any parallel processing. A careful inspection of the CTR Algorithm 1 reveals that it can be easily parallelized by performing the computations for different CV folds on different cores and, even within each CV fold, performing all the inner-product computations in Algorithm 1 for blocks of rows sent to different cores. The `glmnet` algorithm can be similarly parallelized.

OSCAR and fused lasso (Ulbricht, 2012) are orders-of-magnitude more computationally expensive to fit with large p and n. Moreover, it appears that they were designed for smaller p situations. They did not perform well for the values of p and n in our examples. However, although we omit the results for brevity, we did find that OSCAR and fused lasso had predictive performance comparable to CTR for the same examples considered in (Ulbricht, 2012), which all had $p \leq 40$ and $n \leq 100$. In addition, OSCAR and fused lasso

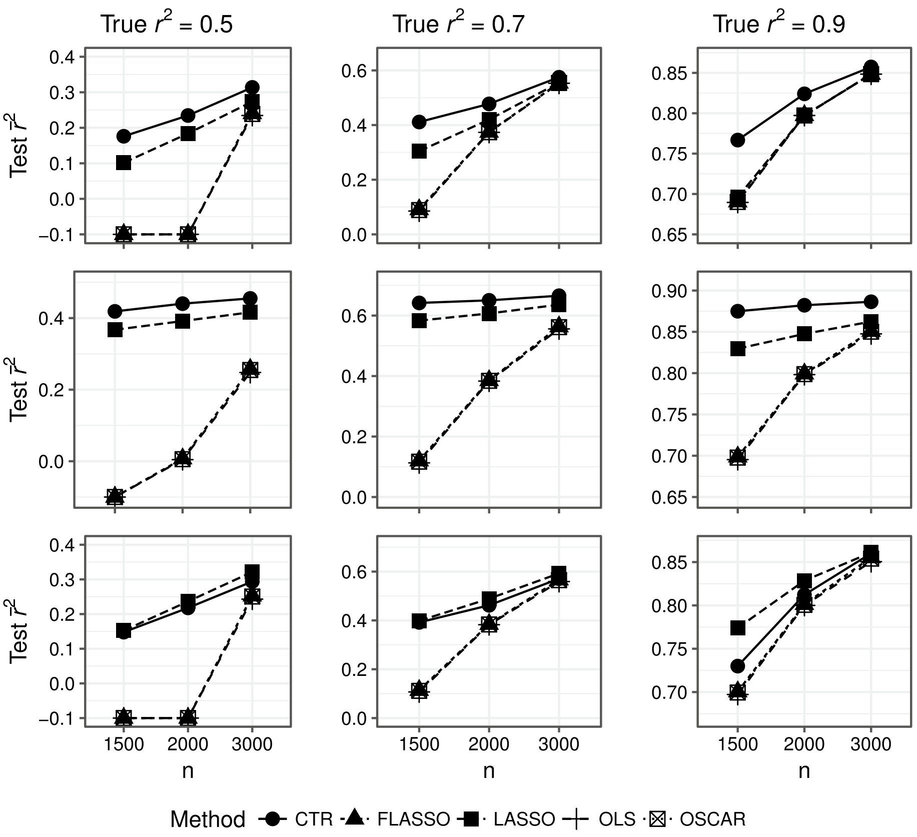


Fig. 7 Simulation results with no true group structure with $\beta_j = (j-p)/100$ for $j = 1, \dots, p/2$, $\beta_j = j/100$ for $j = p/2+1, \dots, p$ and $\mathbf{\Sigma} = \mathbf{I}$ (1st row), $\beta_j = (j-p)/100$ for $j = 1, \dots, p/2$, $\beta_j = j/100$ for $j = p/2+1, \dots, p$ and $\mathbf{\Sigma}_{i,j} = 0.7^{|i-j|}$ (2nd row), $\beta_j = (j-p)/100$ for $j = 1, \dots, p/4$, $\beta_j = 0$ for $j = p/4+1, \dots, 3p/4$, $\beta_j = j/100$ for $j = 3p/4+1, \dots, p$ and $\mathbf{\Sigma} = \mathbf{I}$ (3rd row). Note that LASSO is not a group method and is only included as a reference for comparison

have two tuning parameters in their regularization penalty terms, and their proper selection via CV may be computationally prohibitive considering the computational time to fit a single OSCAR or fused lasso model. In comparison, lasso and CTR each have a single "complexity" tuning parameter (the regularization penalty parameter for lasso, and k for CTR), and their computational advantages allow for much faster parameter selection via CV.

Overall, the excellent predictive performance of CTR in our experiments implies that our greedy heuristic search gives near-optimal solutions that are possibly not far from optimal, in addition to being computationally inexpensive. The CTR test r^2 values in Figs. 4, 5, 6 and 7 for larger n are often close to the true r^2 values, and no model can do better than the true r^2. Even though an optimal group structure with lower SSE could be found with infinite computational resources and an exhaustive search, one can use CTR with outstanding predictive and computational performance to construct interpretable models.

Table 1 Predictive accuracy comparison for CTR and lasso when $n \in \{750, 6000\}$. The group-based methods (OSCAR and fused lasso) had substantially worse accuracy for small n and prohibitive computational expense for large n, and are omitted

n	r^2	1st row		2nd row		3rd row	
		CTR	LASSO	CTR	LASSO	CTR	LASSO
(a) Figure 4 experiments							
750	0.5	0.07	0.01	0.26	0.21	0.26	0.02
750	0.7	0.23	0.06	0.49	0.34	0.44	0.10
750	0.9	0.45	0.20	0.85	0.52	0.71	0.27
6000	0.5	0.43	0.41	0.46	0.42	0.42	0.41
6000	0.7	0.67	0.64	0.69	0.65	0.65	0.64
6000	0.9	0.89	0.88	0.90	0.88	0.88	0.88
(b) Figure 5 experiments							
750	0.5	0.10	0.32	0.20	0.23	0.08	0.03
750	0.7	0.46	0.59	0.43	0.37	0.21	0.11
750	0.9	0.82	0.86	0.68	0.57	0.43	0.28
6000	0.5	0.50	0.49	0.43	0.43	0.41	0.42
6000	0.7	0.70	0.69	0.65	0.66	0.65	0.65
6000	0.9	0.90	0.90	0.90	0.88	0.89	0.88
(c) Figure 6 experiments							
750	0.5	0.08	0.03	0.38	0.30	0.15	0.07
750	0.7	0.20	0.11	0.59	0.52	0.31	0.21
750	0.9	0.43	0.29	0.84	0.78	0.55	0.49
6000	0.5	0.40	0.40	0.48	0.46	0.41	0.43
6000	0.7	0.64	0.64	0.69	0.67	0.65	0.66
6000	0.9	0.87	0.88	0.89	0.88	0.88	0.88

Table 2 Computational time (s) comparison of CTR, lasso, OSCAR, and fused lasso for various p and n

p (Table 2a)/n (Table 2b)	CTR		Lasso		OSCAR		FLasso	
	Time	$\bar{r}^2$	Time	$\bar{r}^2$	Time	$\bar{r}^2$	Time	$\bar{r}^2$
(a) Computational time (s) for $n = 10^4$ and $p \in \{500, 1000, 2000, 4000\}$								
500	4	0.89	29	0.89	4363	0.89	4446	0.89
1000	8	0.89	55	0.89	8704	0.89	8853	0.89
2000	16	0.89	107	0.88	10800[1]	–	10800[1]	–
4000	30	0.88	220	0.86	10800[1]	–	10800[1]	–
(b) Computational time (s) for $p = 500$ and $n \in \{10^3, 10^4, 10^5, 10^6\}$								
10^3	0	0.87	4	0.85	274	0.80	342	0.79
10^4	4	0.89	29	0.89	4363	0.89	4446	0.89
10^5	43	0.90	294	0.90	10800[1]	–	10800[1]	–
10^6	577	0.90	4042	0.90	10800[1]	–	10800[1]	–

[1]OSCAR and fused lasso were terminated after 3 h without converging

3.3 Performance comparison on real-world data

In this section, we compare the methods using publicly available real data sets. To do that, we first use the residential building data set (Rafiei and Adeli, 2016) from UCI Machine Learning Repository. The data set consists of 8 physical and financial predictors and 19 economic predictors, each of which is measured at five different time points (thus comprising $5 \times 19 = 95$ predictors observed quarterly prior to the start of the construction), resulting in a total of 103 predictors. We first fit all methods using all $p = 103$ predictors to predict the response variable, construction cost. The data set consists of 372 observations, and we split the data into training and test sets, where we vary the size of the training set as 74, 186, 298 and the size of the test set is fixed as 74 to examine the effect of the amount of training data on out-of-sample performance. We repeated this process for 100 different splittings of the data into training and test sets of respective sizes, and the average test $\bar{r}^2$ is given in Fig. 8 (left). Regarding the predictive accuracy, with the training sample size of 298 all methods perform well with the test $\bar{r}^2$ around 0.96, with CTR having slightly better performance than the other group-based methods and lasso. The difference between methods is more distinguishable for the smaller sample sizes, and in this case CTR performed better than the other group-based methods. Overall, CTR performed only slightly better than lasso (slightly worse for the smallest sample size and slightly better for the medium and larger sample sizes). Regarding interpretability, among eight original physical and financial predictors CTR includes only a single predictor "Preliminary estimated construction cost based on the prices at the beginning of the project" into the first group with a very large positive coefficient. This makes sense because one would expect the preliminary estimated cost to play a key role in determining the actual construction cost. The second CTR group includes the single predictor "Duration of construction" with a relatively large positive coefficient as well, and this implies that the actual cost is likely to increase with an increasing duration of construction. After this point, the subsequent CTR groups start to include lagged economic predictors. For example, the fourth group includes predictors measured at lagged time points three, four and five, indicating that this group includes the predictors that are influential at earlier time points. Based on these results, original eight physical and financial predictors explain most of the variability existing in the data set, and thus the test $\bar{r}^2$ is very close to one. In order to make the analysis more challenging and provide a different comparison of the predictive accuracy of the various methods, we exclude the original eight project physical predictors from the models, and refit the models with the remaining 95 economic predictors. The results are given in Fig. 8 (right). In this case, the relative performance of CTR and lasso is similar to the case when all of the original predictors are included. However, the performance difference between CTR and the other group-based methods is more pronounced. For example, for the smallest training sample size, CTR achieves a test $\bar{r}^2 = 0.56$, whereas the other group-based methods only have test $\bar{r}^2 = 0.25$ approximately.

Next, we consider monthly natural gas prices from the U.S. Energy Information Administration between January 1997 and August 2020 (the data set is available at (DataHub, 2021)). We consider the regression model

$$\hat{x}_{i+L} = \hat{\beta}_1 x_i + \hat{\beta}_2 x_{i+1} + \cdots + \hat{\beta}_L x_{i+L-1}, \text{ for } i = 1, \dots, n-L, \tag{18}$$

where L represents the maximum time lag in the model (we consider various values of L below) and $n = 284$ is the number of observations. We expect that there should be an underlying group structure existing in these types of regression models, and predictors at

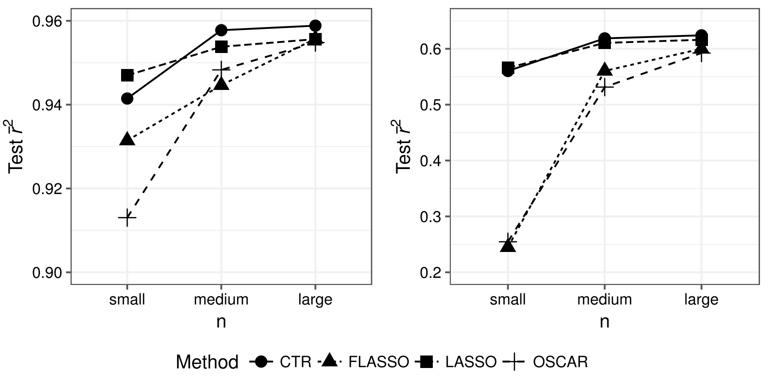


Fig. 8 Residential building data set results with $p = 103$ (left) and $p = 95$ (right)

the most recent time points would have more influence on the response variable. In order to check the performance of each method, the data are split 50:50 into training and testing set, where the latter includes observations at the most recent time points. Figure 9 (left) illustrates the predictive performance as L is varied, from which we see that CTR performs better than the other methods, substantially so for the larger L values. Finally, in order the compare the performance with larger p, we consider the daily volatility index data set from 2004 to 2011, which can be found at (DataHub, 2021). We fit a regression model similar to (18) with varying training sizes 750, 1000 and 1850 with $p = 500$, the results for which are given in Fig. 9 (right). Overall, CTR performs better than OLS and the other group-based methods (substantially so for the small and medium sample sizes) and comparably to lasso.

Moreover, we compared the predictive accuracy and computational performance using a number of other data sets available at the UCI machine learning repository: superconductivity (with $p = 81$, $n = 21263$), communities and crime (with $p = 128$, $n = 1994$), wine (with $p = 12$, $n = 4898$) and yacht hydrodynamics (with $p = 7$, $n = 308$). For which, we consider 100 different random 80:20 splittings into training and test sets. All methods performed comparably in terms of predictive accuracy with the resulting test $\bar{r}^2$ values being approximately 0.65, 0.73, 0.34, 0.27, and 0.64 for the superconductivity, communities and crime, wine (red), wine (white), and yacht hydrodynamics data sets, respectively. This may be because these real data sets may not truly have a group structure, and with sufficient number of observations all methods perform similarly to each other. It is important to note that although CTR has comparable predictive accuracy for these five data sets, it is orders-of-magnitude faster than the other group-based methods and even substantially faster than lasso.

3.4 Feature selection via CTR

This section investigates the effectiveness of the methods in terms of accurately identifying zero- and nonzero-coefficient predictors. To do this, we use a simulation experiment with a true group structure $\boldsymbol{\beta}^\top = \left[\mathbf{1}_{0.3p}^\top, \mathbf{0}_{0.4p}^\top, -\mathbf{1}_{0.3p}^\top\right]$ with $\boldsymbol{\Sigma} = \mathbf{I}$ and $p = 100$. Thus, we have 40 inactive predictors with a coefficient of zero (i.e., a zero-coefficient group with a size of 40) and 60 nonzero-coefficient predictors. We evaluate the performance of CTR, fused lasso, lasso, OLS and OSCAR by the number of falsely identified predictors of both types, i.e., the number of truly zero-coefficient predictors whose estimated coefficients differ from

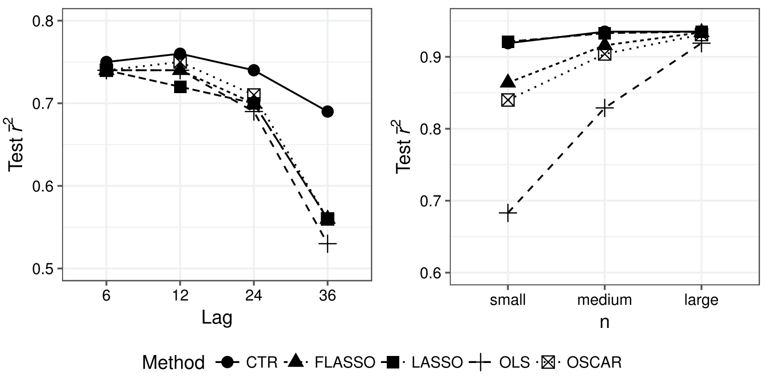


Fig. 9 Monthly natural gas prices (left) and daily volatility index (right) data sets results

zero and the number of truly nonzero-coefficient predictors whose estimated coefficients are zero. For the comparison, we generated a test set of 10,000 observations, and 100 training sets (corresponding to 100 replicates), each of size $n \in \{150, 300, 600\}$, similar to the experiments in Sect. 3.1. In addition, we varied the true $r^2 \in \{0.5, 0.7, 0.9\}$ to represent different noise levels.

Figure 10 shows the predictive performances of all methods, which are in line with the ones presented in Sect. 3.1. The rows of Fig. 11 show the distribution of falsely identified predictors, falsely identified zero-coefficients, and falsely identified nonzero-coefficients, respectively, across the 100 replicates for both CTR and lasso. The results for the remaining methods are not included in this plot since they are not able to exclude any zero-coefficient predictor from any of 100 replicates. Overall, CTR is no more nor less subject to falsely identified coefficients than lasso, and when n is large, CTR is able to identify the predictors more accurately.

Next, we analyze the bias in the coefficients. To illustrate this, Fig. 12 shows the mean and the standard error of the estimated coefficients across 100 replicates when the true $r^2 = 0.9$ and $n = 600$ for all methods (we use this case because the predictive accuracy is high for all methods). With CTR, even though zero-coefficient predictors are sometimes included in the model, they are included in a group with a coefficient close to zero. Moreover, both lasso and fused lasso shrink the coefficients towards zero, whereas the bias introduced by CTR is close to zero in Fig. 12. In order to further demonstrate the difference in bias between CTR and lasso for a more challenging case, Fig. 13 shows analogous results for $n = 150$ and the true $r^2 = 0.5$.

3.5 Stability of CTR

As mentioned above, the split-point search procedure is analogous to how splits are chosen greedily when fitting standard regression trees. However, CTR and standard regression trees are different in that CTR splits a set of predictors into two subgroups of predictors, whereas regression trees split an interval of values of a single predictor into two subintervals. Standard classification and regressions trees are known to be unstable procedures (Breiman, 1996; Hastie et al., 2009), in the sense that the split points are determined by the observations in the training data, and slight modifications in the training data result in

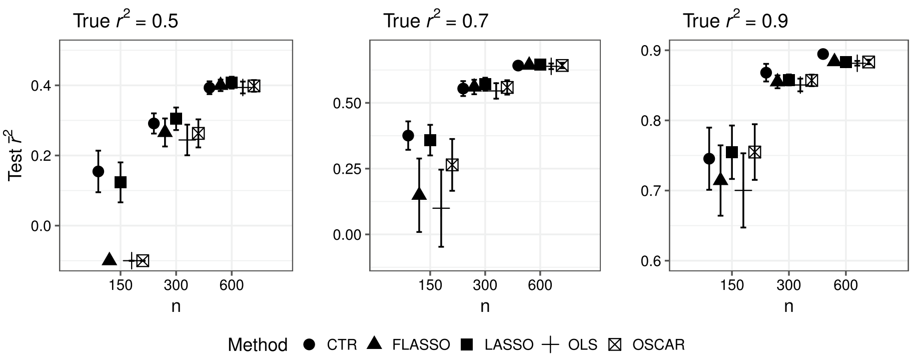


Fig. 10 Simulation results with true group structure $\boldsymbol{\beta}^{\mathsf{T}} = \left[\mathbf{1}^{\mathsf{T}}_{0.3p}, \mathbf{0}^{\mathsf{T}}_{0.4p}, -\mathbf{1}^{\mathsf{T}}_{0.3p}\right]$ with $\boldsymbol{\Sigma} = \mathbf{I}$ and $p = 100$. The error bars indicate the standard deviation of the test r^2 across the 100 replicates

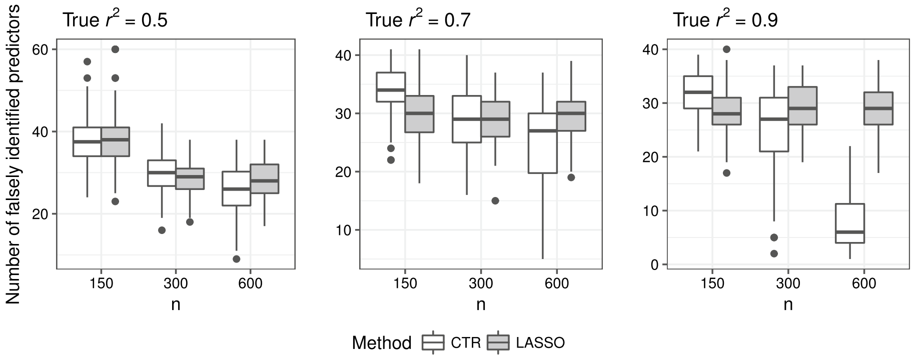
a completely different tree structures. In contrast, in CTR, the decision to split the predictors into the groups depends on the marginal reduction in SSE for each predictor, and even if we change the training data slightly, the marginal reduction in SSE will not be affected substantially, as the remaining observations in the data will continue to contribute to the marginal reduction in SSE.

In order to demonstrate the instability of standard classification/regression trees, Hastie et al. (2009) generated bootstrap samples from the training data and refit the trees to each of the bootstrap data sets. Following this procedure, we generated 100 bootstrap data sets, and refit the model to each of the bootstrap data sets for each of the methods used in benchmark studies to examine whether the methods are sensitive to these changes in the data sets. We use a simulation experiment with a true group structure $\boldsymbol{\beta}^{\mathsf{T}} = \left[\mathbf{5}^{\mathsf{T}}_{p/5}, \mathbf{4}^{\mathsf{T}}_{p/5}, \mathbf{3}^{\mathsf{T}}_{p/5}, \mathbf{2}^{\mathsf{T}}_{p/5}, \mathbf{1}^{\mathsf{T}}_{p/5}\right]$ with $\boldsymbol{\Sigma} = \mathbf{I}$ and $p = 100$. For the comparisons, we generated a test set of 10,000 observations, and 100 training sets of size $n \in \{300, 600\}$ (note that we did not use $n = 150$ because no methods performed well with bootstrap samples when original training sample size is 150). In addition, we varied the true $r^2 \in \{0.5, 0.7, 0.9\}$ to represent different noise levels.

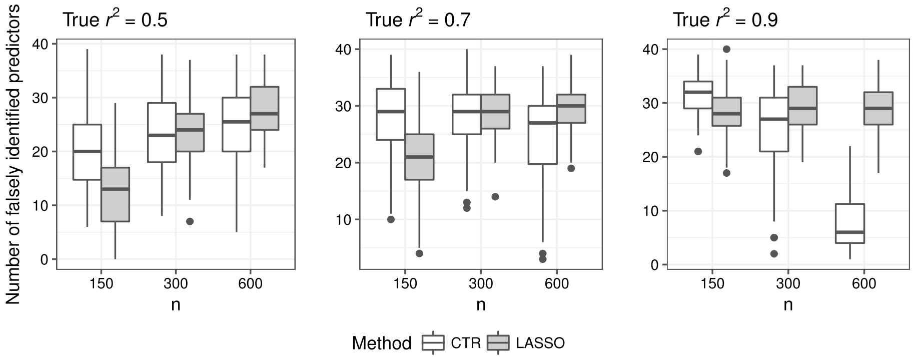
Figure 14 illustrates the variability in the predictive performance of each of the methods with different sample sizes and noise levels. The accuracy of each method does not vary significantly across different bootstrap samples. In addition, Fig. 15 shows the mean and standard error of the estimated coefficients across the 100 replicates for CTR and lasso. The results demonstrate that CTR is no less stable than lasso, in spite of its use of trees to find the group structure. Note that we only provide results for when $n = 600$ and true $r^2 = 0.9$ for illustration, and the CTR results are no less stable than the other methods in the other cases.

4 Discovering hidden ontologies: a programmatic advertising example

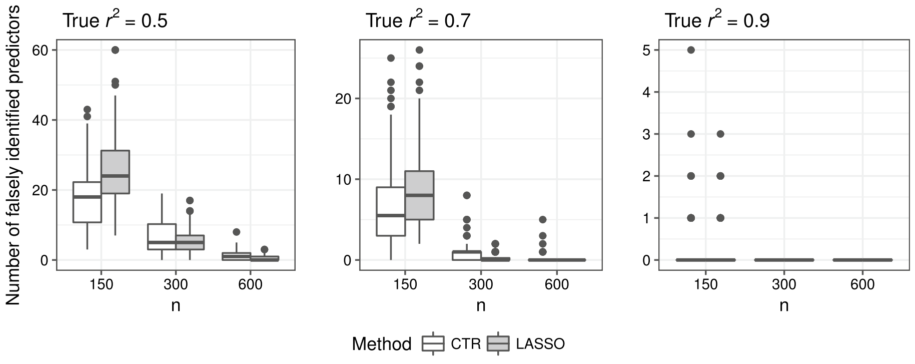
We now give a programmatic advertising example using real data to illustrate how the group structure of the derived features from CTR can be quite interpretable and can help to discover hidden ontologies that provide insight into the regression dependencies. Programmatic advertising delivers messages to individuals, where the advertisers' bids are



(a) Number of falsely identified predictors



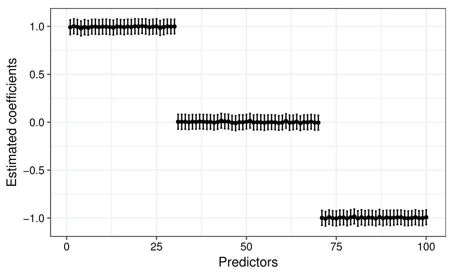
(b) Number of falsely identified zero-coefficient predictors



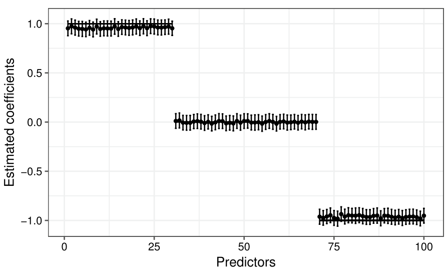
(c) Number of falsely identified nonzero-coefficient predictors

Fig. 11 Feature selection performance for CTR and lasso for the experiments with true group structure $\boldsymbol{\beta}^{\top} = \left[\mathbf{1}_{0.3p}^{\top}, \mathbf{0}_{0.4p}^{\top}, -\mathbf{1}_{0.3p}^{\top}\right]$ with $\boldsymbol{\Sigma} = \mathbf{I}$ and $p = 100$

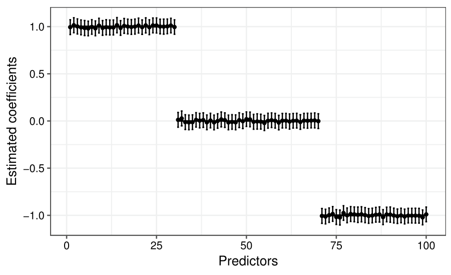
informed by learning algorithms that associate what is known about a consumer with the consumer's interest in the advertised product or service (Perlich et al., 2014). With proliferation of programmatic ad exchange markets, it has been used in many different forms of advertising (e.g., search, display and video) to make data-driven decisions. As a political



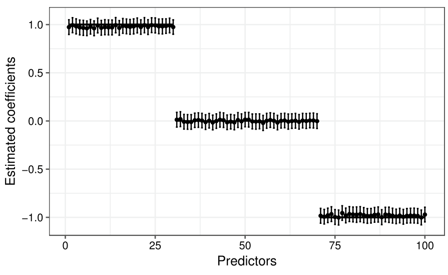
(a) CTR.



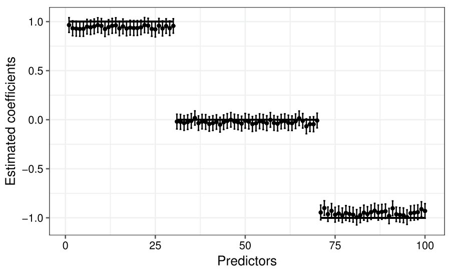
(b) LASSO.



(c) OLS.



(d) OSCAR.

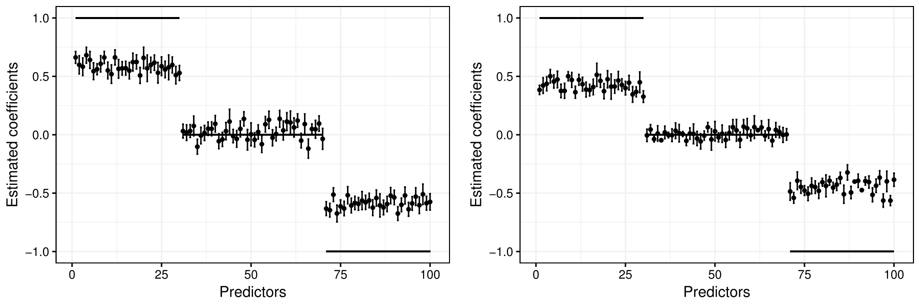


(e) FUSED LASSO.

Fig. 12 Estimated coefficients for the experiment with $\boldsymbol{\beta}^{\top} = \left[\mathbf{1}_{0.3p}^{\top}, \mathbf{0}_{0.4p}^{\top}, -\mathbf{1}_{0.3p}^{\top}\right]$ with $\boldsymbol{\Sigma} = \mathbf{I}$ and $p = 100$ and $n = 600$. The error bars indicate one standard error of estimated coefficients across the 100 replicates. The horizontal black line illustrates the true coefficients of the corresponding predictors

advertising example, consider that during the 2016 election cycle, \$9.8 billion dollars were spent on political campaign advertising, and programmatic advertising to target specific types of voters was an important driver of this amount (Kaye, 2017).

In this section we consider such an example and use CTR to predict political leanings (the response) based on TV viewing habits (the predictors) of households. The example involves joining two data sets: cable TV set-top-box (STB) data from April to December 2015 recording everything households watch on TV, and household voting data for the major political parties' 2016 primary elections, both of which are for a single state. Publicly available records indicate which registered voters voted in the 2016 primary elections, and, if so, whether they voted in the Democratic or Republican primary (note that the data only indicates whether the voter voted in the primary and not for which candidate he/she voted). The cable TV provider matched the two data sets on household address with 47,979



(a) CTR. **(b)** LASSO.

Fig. 13 Estimated coefficients for the experiment with $\boldsymbol{\beta}^{\mathsf{T}} = \left[\mathbf{1}_{0.3p}^{\mathsf{T}}, \mathbf{0}_{0.4p}^{\mathsf{T}}, -\mathbf{1}_{0.3p}^{\mathsf{T}}\right]$ with $\boldsymbol{\Sigma} = \mathbf{I}$ and $p = 100$ and $n = 150$. The error bars indicate one standard error of estimated coefficients across the 100 replicates. The horizontal black line illustrates the true coefficients of the corresponding predictors

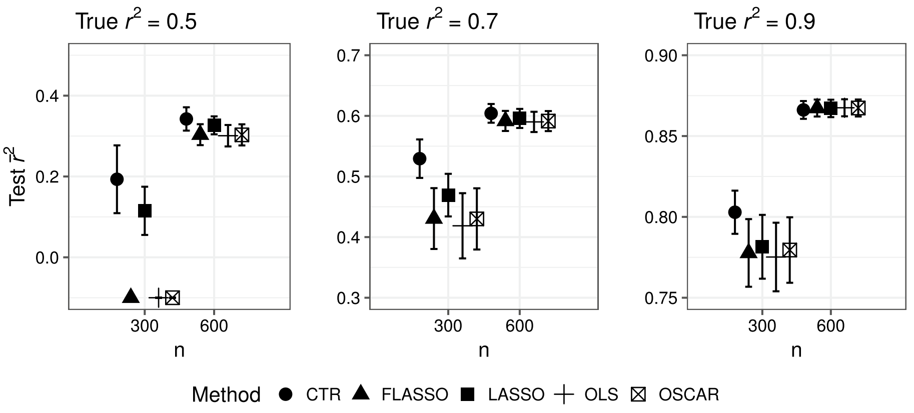
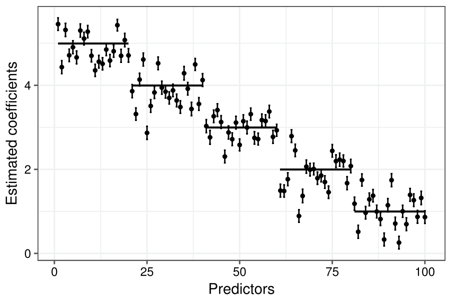


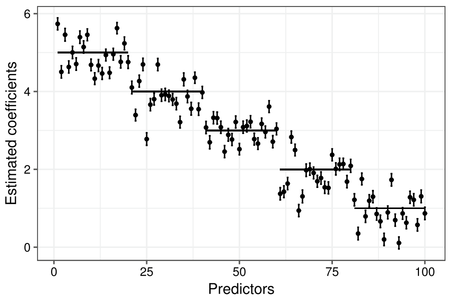
Fig. 14 Simulation results with true group structure $\boldsymbol{\beta}^{\mathsf{T}} = \left[\mathbf{5}_{p/5}^{\mathsf{T}}, \mathbf{4}_{p/5}^{\mathsf{T}}, \mathbf{3}_{p/5}^{\mathsf{T}}, \mathbf{2}_{p/5}^{\mathsf{T}}, \mathbf{1}_{p/5}^{\mathsf{T}}\right]$ with $\boldsymbol{\Sigma} = \mathbf{I}$ and $p = 100$. The error bars indicate the standard deviation of estimated r^2 values across the 100 replicates each of which is fitted to bootstrap samples

matches, although the final data set that we analyze is anonymized in this regard. The final data therefore consisted of TV viewing information for these 47,979 households, along with the number of Democratic and Republican primary votes cast in each household.

The following are details on what constitutes the predictor variables and the response variable in this example. Each "row" of data corresponds to one household for which the voting data was matched with the TV viewing data, and the training set consisted of 15,000 randomly selected rows. The initial data set is moderately unbalanced (the number of households with at least one member who voted in the Democratic primary is 8439, and the rest of the 39,540 households do not have any member who voted in the Democratic primary). To have more balanced data, we sampled 7500 households from each group to create the training set, and used the remaining to construct a test set. The response variable was defined as the fraction of primary voters in each household who voted in the Democrat primary (i.e., Democrat primary votes, divided by the sum of Democrat and Republican primary votes). Because viewership of news (and other) programs is expected to be related



(a) CTR.



(b) LASSO.

Fig. 15 Estimated coefficients for the experiment with $\boldsymbol{\beta}^{\mathrm{T}} = \left[\mathbf{5}_{p/5}^{\mathrm{T}}, \mathbf{4}_{p/5}^{\mathrm{T}}, \mathbf{3}_{p/5}^{\mathrm{T}}, \mathbf{2}_{p/5}^{\mathrm{T}}, \mathbf{1}_{p/5}^{\mathrm{T}}\right]$ with $\boldsymbol{\Sigma} = \mathbf{I}$ and $p = 100$, $n = 600$ and true $r^2 = 0.9$. The error bars indicate one standard error of estimated coefficients across the 100 replicates with bootstrap samples. The horizontal black line illustrates the true coefficients of the corresponding predictors

to political leanings (Ksiazek et al., 2010), over the full April–December, 2015 TV viewing period we calculated the total watch times of 238 distinct news programs, such as *The O'Reilly Factor*, *The Rachel Maddow Show*, etc., some of which are hosted by people with well-known political leanings. For non-news related programming (e.g., *basketball*, *drama*, *reality*, *talk shows*, *religious*, etc.) we created predictors (the original predictors $\mathbf{X}$; not our CTR-derived features $\mathbf{Z}$) that are the total watch times within each genre, as opposed to within each specific program. In total, we created 288 viewing predictors that each represent the total watch time of some show or genre of shows. The predictor variables were logged prior to the analysis.

We then applied the CTR algorithm to obtain the group structure. Figure 16 plots the CV SSE / SST versus the number of derived features ranging from 1 to 20. The best number of derived features was estimated to be $k = 12$ according to 10-fold CV, and the final group structure with $k = 12$ derived features is given in Table 3. Note that the improvement in CV r^2 (CV $r^2 = 1 -$ CV SSE / SST) was small beyond three groups. We therefore include only the first three groups in Fig. 17 for viewing convenience and interpretability purposes. The plot similar to Fig. 16 allows users to select k to balance between better model simplicity/interpretability and better predictive accuracy in terms of CV r^2, depending on their particular needs and plans for using the regression model.

Figure 17 shows that during iteration $k = 1$, CTR extracted a derived feature with a negative coefficient $\hat{\alpha}_{1,1} = -4.90$. The group $G_{1,1}$ consisted of programs on the Fox News Channel such as *America's Newsroom*, *Fox & Friends*, *Bret Baier*, *Outnumbered*, *Varney & Co.*, *The Five*, *The O'Reilly Factor* and individual programs *Golf Central* and *Duck Dynasty*. For this first group, the CV $r^2 = 0.15$. The remaining programs were placed into the zero-coefficient group $G_{1,2}$ after iteration $k = 1$. The negative coefficient $\hat{\alpha}_{1,1} = -4.90$ makes sense, since a higher response means a more Democrat-leaning voting pattern, and the Fox news, *Golf Central*, and *Duck Dynasty* programs in $G_{1,1}$ are widely regarded as conservative oriented.

During iteration $k = 2$, CTR found a new group of predictors by splitting the zero-coefficient group $G_{1,2}$ into a positive-coefficient group $G_{2,2}$ with $\hat{\alpha}_{2,2} = 1.60$ and a zero-coefficient group $G_{2,3}$. The first group stayed the same (i.e., $G_{2,1} = G_{1,1}$), although its coefficient was updated ($\hat{\alpha}_{2,1} = -6.20$). The CV r^2 value increased to 0.30 with the second derived

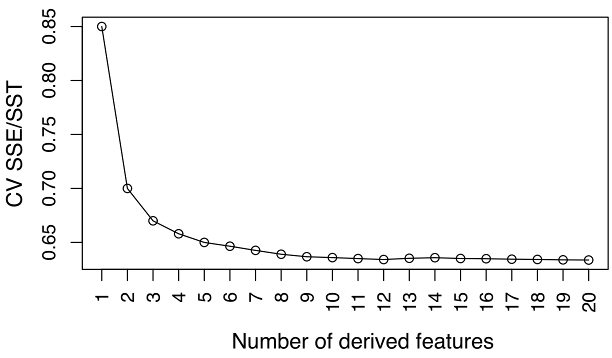


Fig. 16 CV SSE/SST for the programmatic advertising example with the number of derived features ranging from 1 to 20

feature. The programs in $G_{2,2}$ included some from CNN (e.g., *Wolf Blitzer*, *Ashleigh Banfield*, *Erin Burnett*, *New Day*, *Smerconish*, *Jake Tapper*), some from MSNBC (e.g., *Alex Witt*, *Andrea Mitchell*, *All In with Chris Hayes*, *Hardball*, *Harris Perry*, *The Last Word*, *Live*, *Rachel Maddow* and *Rundown*) and several news programs from broadcast networks (*Spanish News*, *The View*, *CBS Overnight News*, *Right This Minute*, and *Public TV News*). Most of these news shows are widely regarded as liberal leaning, in contrast to the conservative leaning news shows in $G_{2,1}$. The genres of *community*, *horror*, and *interview biographies*, as well as the individual programs *Charmed*, *Keeping up with the Kardashians*, *Forensic Files*, *Steve Harvey*, *Teen Titans Go!*, *Supernatural*, *Maury*, *Snapped* and *The People's Court*, are included in $G_{2,2}$, along with the liberal-leaning news shows.

During iteration $k = 3$, a new group of predictors $G_{3,3}$ entered the model with a negative group coefficient $\hat{\alpha}_{3,3} = -1.40$. The third derived feature included the genres *golf*, *baseball*, *Nascar*, *history* and *romance*, some programs from The Fox Channel (e.g., *Hannity*, *Happening Now*, *Greta Van Susteren*, *The Kelly File*, *Chris Wallace*), some individual programs *NCIS*, *Three House Masters*, *OK TV*, *American Pickers*, *Shark Tank*, *Fixer Upper*, *Alaskan Bush People*, *Days of Our Lives*, *The Waltons* and *America's Got Talent*, along with some finance news (such as *Fast Money*, *Power Lunch*, *Squawk Box*). Including this third derived feature increased the CV $r^2 = 0.33$. The first and third derived features had negative coefficients $\hat{\alpha}_{3,1} = -3.80$ and $\hat{\alpha}_{3,3} = -1.40$, whereas the second derived feature had a positive coefficient $\hat{\alpha}_{3,2} = 2.20$.

The group structure provides insight into the relationship between political affiliations and television viewing habits. For example, the first two derived features are not surprising and are consistent with the conservative-leaning or liberal-leaning reputations of the various news shows. In the third group, the genres golf, Nascar and baseball, which have larger average viewing ages,[2] are sometimes associated with Republicans (Maniam and Smith 2014). Some of the TV shows listed in the third derived feature have been discussed in this context in the popular press (Gay, 2012).

The CV r^2 of 0.33 might be viewed as low in some contexts. However, in applications like this, an r^2 of 0.33 may be quite useful practically. To assess the utility of the model in such situations, one often uses a cumulative gains chart (Larose, 2005). Figure 18 shows the cumulative gains chart for this example, which we calculated as follows. We first used our final fitted CTR model (with three derived features) to estimate the fraction of

[2] See https://www.marketwatch.com/story/the-sports-with-the-oldest-and-youngest-tv-audiences-2017-06-30.

Table 3 Final derived features of the CTR-estimated group structure and coefficients in the programmatic advertising example

Group	Predictors	α
1	*Fox & Friends*, *Bret Baier*, *The O'Reilly Factor*	− 4.3
2	*America's Newsroom*,*The Five*, *Outnumbered*, *Varney & Co* *Golf Central*	− 0.7
3	*All In with Chris Hayes*, *Hardball*, *The Last Word*, *Rachel Maddow*, *The View*, *Maury*, *Snapped*, *Steve Harvey*, *Public TV News*, *community*, *interview biographies*	3.6
4	*Alex Witt*, *Andrea Mitchell*, *Harris Perry*, *Rundown*, *Live*, *CBS Overnight News*	− 0.6
5	*Wolf Blitzer*, *Ashleigh Banfield*, *Erin Burnett*, *New Day*, *Smerconish*, *Jake Tapper*, *Forensic Files*	1.3
6	*Duck Dynasty*	− 8.9
7	*Alaskan Bush People*, *golf*, *baseball*, *Nascar*, *history*, *American Pickers*, *Shark Tank*, *Fixer Upper*	− 2.8
8	*Teen Titans Go!*, *Spanish News*, *Keeping up with the Kardashians*, *The People's Court*, *horror*, *Charmed*, *Supernatural* *Right This Minute*	1.3
9	*Big Bang Theory*, *Friends*, *Seinfeld*, *HIMYM*, *Rules Of Enagement*	− 1.6
10	*America This Morning*, *AgDay*, *CBS Morning News*, *Crime Watch Daily*, *Red Eye*, *Shepard Smith*, *Jeopardy*, *Let's Make a Deal*, *Log Cabin Living*, *The Young and the Restless*, *House Hunters International*, *Matlock*, *Property Brothers*, *The Talk*, *Love It Or List It*, *The Bold And The Beautiful*	1.5
11	*romance*, *Hannity*, *Happening Now*, *Greta Van Susteren*, *The Kelly File*, *Chris Wallace*, *NCIS*, *Three House Masters*, *OK TV*, *Days of Our Lives*, *The Waltons*, *America's Got Talent*, *Fast Money*, *Power Lunch*, *Squawk Box*	− 1.2
12	*ABC World News Tonight*, *Baseball Tonight*, *CBS News*, *Modern Family*, *Tonight*, *Bones*, *The King of Queens*, *Law and Order*, *Jimmy Kimmel*, *Ellen DeGeneres*, *Dateline NBC*, *Stephen Colbert*, *Inside Edition*, *The Doctors*, *Weekend Recharge*, *Dr. Phil*	0.6

Democrat votes in each household (the response variable in our model), and we then sorted households in descending order according to those values. For each fraction between 0 and 1 (which is the horizontal-axis value in Fig. 18), we then calculated the "fraction of Democrat votes reached," which is defined as follows. For each specified fraction between 0–1, we took that fraction of top households as ordered by the predicted responses from our CTR model, and we counted the total number of actual Democrat votes in those households. This number, divided by the total number of actual Democrat votes in all household, is defined as the "fraction of Democrat votes reached" and is plotted on the vertical axes

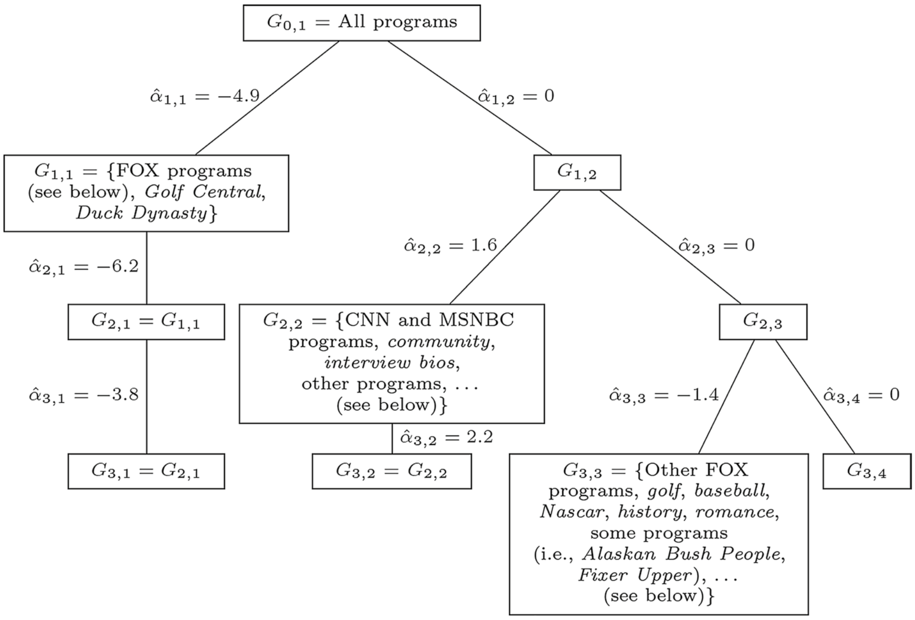

$G_{0,1}$ = All programs
$\hat{\alpha}_{1,1} = -4.9$
$\hat{\alpha}_{1,2} = 0$
$G_{1,1}$ = {FOX programs (see below), *Golf Central*, *Duck Dynasty*}
$G_{1,2}$
$\hat{\alpha}_{2,1} = -6.2$
$\hat{\alpha}_{2,2} = 1.6$
$\hat{\alpha}_{2,3} = 0$
$G_{2,1} = G_{1,1}$
$G_{2,2}$ = {CNN and MSNBC programs, *community*, *interview bios*, other programs, ... (see below)}
$G_{2,3}$
$\hat{\alpha}_{3,1} = -3.8$
$\hat{\alpha}_{3,2} = 2.2$
$\hat{\alpha}_{3,3} = -1.4$
$\hat{\alpha}_{3,4} = 0$
$G_{3,1} = G_{2,1}$
$G_{3,2} = G_{2,2}$
$G_{3,3}$ = {Other FOX programs, *golf*, *baseball*, *Nascar*, *history*, *romance*, some programs (i.e., *Alaskan Bush People*, *Fixer Upper*), ... (see below)}
$G_{3,4}$

Fig. 17 Illustration of the CTR-estimated hierarchical group structure and coefficients in the programmatic advertising example

in Fig. 18. The 45-degree line represents the fraction of Democrat votes reached if one randomly selected the households to which to advertise. Thus, the area between the two curves represents the gains in the number of Democrat votes reached if one advertised to the households having the highest CTR-predicted response, relative to randomly choosing households for advertising. Even though the r^2 is only 0.33, the gains are sufficient to have practical utility in many programmatic advertising applications.

Regarding the interpretability, the CTR model was parsimonious with only three derived features (each of which being a straight sum that represents the total number of hours watched for the programs in the group), and each had a clear interpretation. In order to gain insight into the predictive performance of the final model with $k = 12$ derived features, we created a test set of 1,878 households, and compared the test set prediction results with the same benchmark methods considered in Sect. 3. The r^2 values were 0.3550, 0.3573, 0.3584, 0.3597, and 0.3592 for CTR, OLS, lasso, fused lasso, and OSCAR, respectively. While the predictive performance of the methods is close to each other, CTR had far lower computational expense and clear interpretability. In addition, the hierarchical nature of the CTR tree provides insight into which groups are the most important, based on their position within the tree (the higher up the group, the more important it is).

5 CTR as an approximation to OLS

The results in the previous section demonstrate that, regardless of whether there is true group structure, CTR often achieves substantially better predictive accuracy than OLS when the number k of groups is chosen via CV. One way to view choice of k in CTR is that

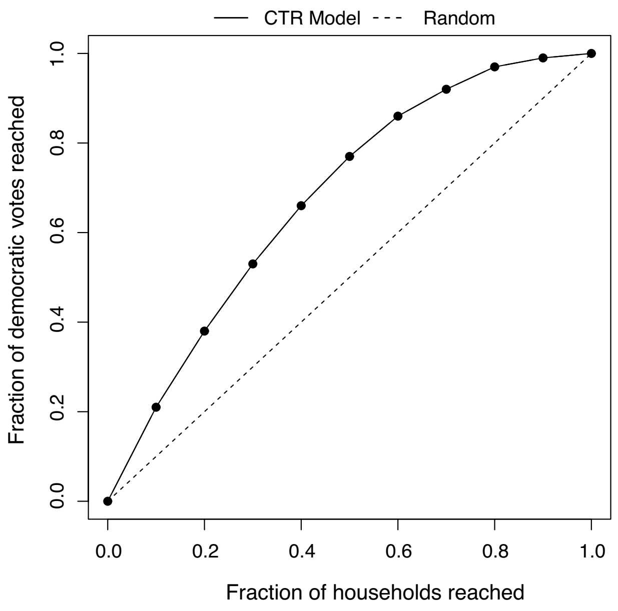


Fig. 18 Gains chart for the programmatic advertising example

we stop at the value for which we conclude (according to CV) that we have built the best model possible, and the CTR model with $k < p$ is as an approximation to OLS, where the approximation can be made arbitrarily close to OLS as k and k_{max} increases. Indeed, in the hypothetical limiting case, when $k = p$, the CTR model is identical to the OLS model. In this section, we further explore CTR as an approximate implementation of OLS that scales up computationally much better than OLS for large p (in addition to being more accurate).

One would expect situations like in the experiments in Fig. 6 to be the most challenging for CTR, because the coefficients $\{\beta_j : j = 1, 2, \ldots, p\}$ vary continuously with no group structure. However, even in this situation, CTR performs substantially better than OLS. This is especially true for smaller n, because OLS tends to overfit due to having many more coefficients to estimate. The reason CTR still performs well when the coefficients vary smoothly has to do with the nature of the CTR approximation of the coefficients, and this relates closely to our earlier recommendation that $k_{max} = 20$ is generally sufficient. With smoothly varying coefficients, the CTR approximation can be viewed as a piecewise constant approximation (with up to k_{max} pieces) of the monotonically increasing function that represents the ordered coefficients $\{\beta_j : j = 1, 2, \ldots, p\}$ as a function of j. This is illustrated for $p = 500$ in Fig. 19, which plots the ordered coefficients $\{\beta_j : j = 1, 2, \ldots, p\}$ for the situations that the coefficients are generated randomly from a Unif$(-1, 1)$ distribution (left panel) and N$(0, 1)$ distribution (right panel). Also plotted are piecewise constant approximations of the coefficients using 20 pieces each (since we only have the coefficients and no data, the approximations were obtained by fitting a standard regression tree to the ordered coefficients $\{\beta_j : j = 1, 2, \ldots, p\}$ as a function of j). In both cases, the piecewise constant functions approximate the coefficients quite closely, with training r^2 values (for approximating the ordered coefficients $\{\beta_j : j = 1, 2, \ldots, p\}$, as a function of j) of 0.9974 and 0.9943 for the left and right panels, respectively. This implies that the group coefficient

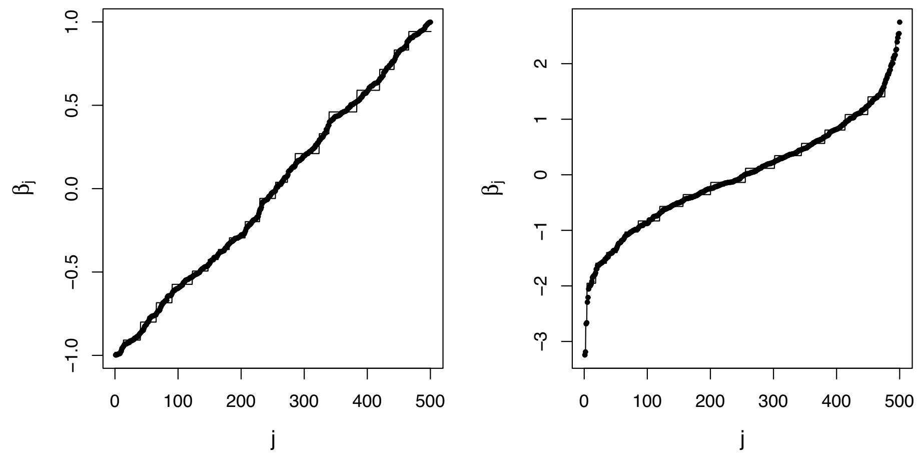


Fig. 19 CTR piecewise constant approximation of $p = 500$ ordered regression coefficients using 20 groups/pieces. Here, $\{\beta_j : j = 1, 2, \dots, 500\} \sim \text{Unif}(-1, 1)$ (left panel) and $\{\beta_j : j = 1, 2, \dots, 500\} \sim \text{N}(0, 1)$ (right panel)

structure of CTR with $k_{max} = 20$ should be sufficient to accurately represent the true coefficients, even when they vary smoothly.

More generally, with either a true coefficient group structure or smoothly varying true coefficients, one can view CTR as a form of regularization (like lasso) but without shrinkage to zero (unlike lasso). That is, imposing the CTR group structure and requiring that the estimated coefficients are the same value $\hat{\alpha}_j$ within groups is a form of regularization that discourages the large/erratic values for the estimated coefficients that can occur in OLS when n is not sufficiently large (relative to p) and/or when strong multicollinearity is present. Whereas lasso discourages large/erratic values by shrinking all coefficients toward zero, which can substantially bias coefficients having large true values, CTR accomplishes this by shrinking or growing coefficients within groups to some common value $\hat{\alpha}_j$. As illustrated in Fig. 19, for k_{max} sufficiently large, the bias introduced by this CTR regularization should be acceptable.

In addition to being more accurate than OLS in situations in which regularization is helpful, CTR can be orders of magnitude faster than OLS for large-scale data sets when a "brute-force" application of OLS is used (via computing the matrix and vector multiplications $\mathbf{M} = \mathbf{X}^{\mathrm{T}}\mathbf{X}$ and $\mathbf{n} = \mathbf{X}^{\mathrm{T}}\mathbf{y}$). Theoretically, computation of $\mathbf{M}$ and $\mathbf{M}^{-1}\mathbf{n}$ for OLS is $O(np^2)$ and $O(p^3)$, respectively, whereas inspection of the CTR algorithm reveals it is $O(npk_{max})$ for fixed k_{max}. Consequently, CTR scales up far better than OLS for large p, in addition to being more accurate. Of course, with very large p, one should not use brute-force OLS. The `glmnet` package with $\lambda = 0$ can be viewed as another $O(np)$ OLS approximation that uses an iterative optimization procedure with clever computational tricks to fit the model.

6 Conclusion

We have introduced a new algorithm called coefficient tree regression (CTR) that efficiently discovers the group structure among predictors, engineers derived features accordingly, and fits the resulting regression model in situations in which no prior knowledge of the group structure is available. It has widespread applicability for many high-dimensional regression problems in which there are a large number of predictors but many of them share the same (or very nearly the same) coefficients. CTR graphically displays the estimated model as a tree, and we demonstrated with the programmatic advertising example that it can provide a clear and concise interpretation of the regression relationship. Simulation studies under different scenarios demonstrated its excellent performance in terms of computing time and predictive accuracy. In particular, its computational performance is multiple orders-of-magnitude better (Table 2), and its predictive performance is also dramatically better (Figs. 4, 5, 6, 7), than the existing group methods (fused lasso and OSCAR). Even relative to the non-group-based lasso method, which was included as a reference benchmark for comparison, CTR's computational performance is nearly an order-of-magnitude better, and its predictive performance is better (when there is true group structure with no sparsity) or comparable (when there is sparsity). This was the case even when there was no true group structure and the coefficients varied smoothly.

As future research, we are extending CTR to generalized linear models (GLMs) (Sürer et al., 2021). We are also investigating different application areas in which we can exploit the context of the data to improve accuracy and substantially improve computational expense for extremely large data sets. In particular, we are investigating an extension to longitudinal data over space and/or time, in part because the largest data sets are often longitudinal data, and it is crucial to obtain efficient, interpretable, and accurate models for high-dimensional data sets. There has been studies that exploit the context in the functional data in the prior literature. For example, for support vector machines, Martin-Barragan et al. (2014) propose a generalized regularization term to obtain coefficient functions in different forms (i.e., sparse, smooth, etc.), and Blanquero et al. (2019) define a new kernel function to identify the critical time intervals. Tian and James (2013) propose a method to generate functions with simple structure to represent the functional data in a lower dimensional space. When the data have a spatial and/or temporal component, we often expect that predictor variables that are closer in space and/or time are more likely to be associated with the response a similar manner. For example, in programmatic TV advertising, information regarding how much time individual users spend viewing each TV program or movie at each specific time is available to inform bid amounts for the users. If the predictors consist of the amount of time watched for each program during each small time interval, then the number of predictors explodes. However, a good derived feature could be the total hours spent viewing specific TV programs over some longer (but a priori unknown) time interval, instead of over each small elemental time interval. Because the appropriate time period over which to group the TV show viewing (one day, one week, one month, etc.) is not known a priori, it must be estimated from the data. In such situations, we expect that we can exploit those aspects to facilitate the CTR split-point search procedure and efficiently find the most meaningful derived features.

References

Akaike, H. (1987). *Factor analysis and AIC*. Springer.

Blanquero, R., Carrizosa, E., Jiménez-Cordero, A., & Martín-Barragán, B. (2019). Functional-bandwidth kernel for support vector machine with functional data: An alternating optimization algorithm. *European Journal of Operational Research, 275*(1), 195–207.

Bondell, H. D., & Reich, B. J. (2008). Simultaneous regression shrinkage, variable selection and clustering of predictors with OSCAR. *Biometrics, 64*(1), 115–123.

Breheny, P. (2015). The group exponential lasso for bi-level variable selection. *Biometrics, 71*(3), 731–740.

Breheny, P. H. J. (2009). Penalized methods for bi-level variable selection. *Stat Interface, 2*(3), 369–380.

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*(2), 123–140.

Carrizosa, E., Nogales-Gómez, A., & Romero Morales, D. (2017). Clustering categories in support vector machines. *Omega, 66*, 28–37.

Comon, P. (1992). Independent component analysis. In J. -L. Lacoume (Ed.), Higher-order statistics (pp. 29–38). Elsevier. https://hal.archives-ouvertes.fr/hal-00346684

DataHub. (2021). https://datahub.io

Dettling, M., & Bühlmann, P. (2004). Finding predictive gene groups from microarray data. *Journal of Multivariate Analysis, 90*(1), 106–131.

Donoho, D., & Jin, J. (2008). Higher criticism thresholding: Optimal feature selection when useful features are rare and weak. *Proceedings of the National Academy of Sciences, 105*(39), 14790–14795.

Fan, J., & Li, R. (2006). Statistical challenges with high dimensionality: Feature selection in knowledge discovery (pp. 595–622). In *25th international congress of mathematicians, ICM 2006; Conference date: 22-08-2006 Through 30-08-2006*

Fokoué, E., & Titterington, D. M. (2003). Mixtures of factor analysers. Bayesian estimation and inference by stochastic simulation. *Machine Learning, 50*, 73–94.

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

Gay, V. (2012, August 29). Study determines the TV shows liberals, conservatives watch. *Newsday*. https://www.newsday.com/entertainment/tv/study-determines-the-tv-shows-liberals-conservatives-watch-1.3934869

Hand, D. J. (2006). Classifier technology and the illusion of progress. *Statistical Science, 21*(1), 1–14.

Hastie, T., Tibshirani, R., Botstein, D., & Brown, P. (2000). Supervised harvesting of expression trees. *Genome Biology, 2*, 1–12.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference and prediction* (2nd ed.). Springer.

Hotelling, H. (1936). Relations between two sets of variates. *Biometrika, 28*(3), 321–377.

Huang, J., Ma, S., Xie, H., & Zhang, C. H. (2009). A group bridge approach for variable selection. *Biometrika, 96*(2), 339–355.

Jolliffe, I. (1986). *Principal component analysis*. Springer Verlag.

Kaye, K. (2017). Data-driven targeting creates huge 2016 political ad shift: Broadcast TV down 20%, cable and digital way up. http://adage.com/article/media/2016-political-broadcast-tv-spend-20-cable-52/307346/

Ke, Z. T., Fan, J., & Wu, Y. (2015). Homogeneity pursuit. *Journal of the American Statistical Association, 110*(509), 175–194.

Ksiazek, T. B., Malthouse, E. C., & Webster, J. G. (2010). News-seekers and avoiders: Exploring patterns of total news consumption across media and the relationship to civic participation. *Journal of Broadcasting & Electronic Media, 54*(4), 551–568.

Larose, D. T. (2005). *Discovering knowledge in data*. Springer Publishing Company, Incorporated.

Maniam, S., & Smith, S. (2014). A wider partisan and ideological gap between younger, older generations. http://www.pewresearch.org/fact-tank/2017/03/20/

Martin-Barragan, B., Lillo, R., & Romo, J. (2014). Interpretable support vector machines for functional data. *European Journal of Operational Research, 232*(1), 146–155.

Park, M. Y., Hastie, T., & Tibshirani, R. (2007). Averaged gene expressions for regression. *Biostatistics, 8*(2), 212–227.

Perlich, C., Dalessandro, B., Raeder, T., Stitelman, O., & Provost, F. (2014). Machine learning for targeted display advertising: Transfer learning in action. *Machine Learning, 95*(1), 103–127.

Qiu, D., & Ahn, J. (2020). Grouped variable screening for ultra-high dimensional data for linear model. *Computational Statistics & Data Analysis, 144*, 106894.

Rafiei, M. H., & Adeli, H. (2016). A novel machine learning model for estimation of sale prices of real estate units. *Journal of Construction Engineering and Management, 142*(2), 04015066.

Rudin, C. (2018). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence, 1*, 206–215.

Shen, X., & Huang, H. C. (2010). Grouping pursuit through a regularization solution surface. *Journal of the American Statistical Association, 105*(490), 727–739.

Sürer, Ö., Apley, D. W., & Malthouse, E. C. (2021). Coefficient tree regression for generalized linear models. *Statistical Analysis and Data Mining: The ASA Data Science Journal, 14*, 407–429.

Team RC. (2017). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing.

Tian, T. S., & James, G. M. (2013). Interpretable dimension reduction for classifying functional data. *Computational Statistics & Data Analysis, 57*(1), 282–296.

Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., & Knight, K. (2005). Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67*(1), 91–108.

Ulbricht, J. (2012). lqa: Penalized likelihood inference for GLMs. R package version 1.0-3.

Verbeke, W., Dejaeger, K., Martens, D., Hur, J., & Baesens, B. (2012). New insights into churn prediction in the telecommunication sector: A profit driven data mining approach. *European Journal of Operational Research, 218*(1), 211–229.

Wang, L., Chen, G., & Li, H. (2007). Group SCAD regression analysis for microarray time course gene expression data. *Bioinformatics, 23*(12), 1486–1494.

Yuan, M., & Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 68*(1), 49–67.

Zhao, P., Rocha, G., & Yu, B. (2009). The composite absolute penalties family for grouped and hierarchical variable selection. *The Annals of Statistics, 37*(6A), 3468–3497.

Zhao, S. D., Parmigiani, G., Huttenhower, C., & Waldron, L. (2014). Más-o-menos: A simple sign averaging method for discrimination in genomic data analysis. *Bioinformatics, 30*(21), 3062–3069.

Zhou, N., & Zhu, J. (2010). Group variable selection via a hierarchical lasso and its oracle property. *Statistics and Its Interface, 3*, 557–574.

Zhu, Y., Shen, X., & Pan, W. (2013). Simultaneous grouping pursuit and feature selection over an undirected graph. *Journal of the American Statistical Association, 108*(502), 713–725.