Advances in Data Analysis and Classification
https://doi.org/10.1007/s11634-023-00562-6

REGULAR ARTICLE

Discovering interpretable structure in longitudinal predictors via coefficient trees

Özge Sürer[1] · Daniel W. Apley[2] · Edward C. Malthouse[2]

Received: 28 September 2022 / Accepted: 20 September 2023
© Springer-Verlag GmbH Germany, part of Springer Nature 2023

Abstract
We consider the regression setting in which the response variable is not longitudinal (i.e., it is observed once for each case), but it is assumed to depend functionally on a set of predictors that are observed longitudinally, which is a specific form of functional predictors. In this situation, we often expect that the same predictor observed at nearby time points are more likely to be associated with the response in the same way. In such situations, we can exploit those aspects and discover groups of predictors that share the same (or similar) coefficient according to their temporal proximity. We propose a new algorithm called coefficient tree regression for data in which the non-longitudinal response depends on longitudinal predictors to efficiently discover the underlying temporal characteristics of the data. The approach results in a simple and highly interpretable tree structure from which the hierarchical relationships between groups of predictors that affect the response in a similar manner based on their temporal proximity can be observed, and we demonstrate with a real example that it can provide a clear and concise interpretation of the data. In numerical comparisons over a variety of examples, we show that our approach achieves substantially better predictive accuracy than existing competitors, most likely due to its inherent form of dimensionality reduction that is automatically discovered when fitting the model, in addition to having interpretability advantages and lower computational expense.

Keywords Group structure · Longitudinal predictors · Pattern discovery · Interpretability · Functional data · Sequential data

✉ Özge Sürer
surero@miamioh.edu

Daniel W. Apley
apley@northwestern.edu

Edward C. Malthouse
ecm@northwestern.edu

[1] Department of Information Systems and Analytics, Miami University, 800 E. High Street, Oxford 45056, OH, USA

[2] Industrial Engineering and Management Sciences, Northwestern University, 2145 Sheridan Road, Evanston 60208, IL, USA

Published online: 11 October 2023

Mathematics Subject Classification 62J05 Linear regression; mixed models · 62-08 Computational methods for problems pertaining to statistics · 62H99 None of the above, but in this section

1 Introduction

Longitudinal data sets that involve repeated observations of the same predictors over time and/or space are commonplace in different regression domains due to recent advances in data collection systems. Here, we consider the common situation of longitudinal predictors, whereas the response is not longitudinal. For example, in clinical studies, patients' medical histories on symptom measurements (such as routine blood tests or tumor size), interventions (such as medicine or radiotherapy), and periodic evaluations are consistently collected over time. As another application area, in economic and financial studies, multiple characteristics such as closing price, P/E ratio, and market capacity of each single stock are observed over time. In addition to temporal data, enabled by the internet, sensors, and smartphone technologies, massive amounts of location-based data are being generated and disseminated by billions of people each second. The use of longitudinal data sets leads to massively high-dimensional multivariate regression models because each single predictor (aka, covariate) over time and/or space must be treated as a large number of predictors in the regression model.

For predictive problems involving high-dimensional longitudinal data sets with many predictors, obtaining an interpretable and accurate model in an efficient way is a challenging issue. However, in practice, the predictors that are closer in space and/or time are more likely to be associated with the response in approximately the same way (i.e., they share a common regression coefficient, which is equivalent to the subsequences of predictor observations that are contiguous in time and/or space affecting the response only collectively via their sum), which enables a low-dimensional, group-based representation that overcomes many difficulties of high-dimensional analysis. Since knowledge of the group structure is generally not available in advance, discovering them automatically from the data is fundamentally important to extract new interpretable predictors and to retrieve meaningful and actionable insights from high-dimensional data. Interpreting and visualizing information extracted from high-dimensional data is at the core of data science (Goodman and Flaxman 2017; Rudin 2018; Rai 2020). Models that facilitate interpretation by domain scientists and engineers are important to better understand and design trustworthy systems, especially for decision-critical environments. In this study, we propose a new tree-based method to extract valuable group-based temporal characteristics from longitudinal data sets and fit the subsequent linear regression model. Tree-based approaches have been used in different contexts because they are very interpretable and conceptually simple (see for examples in Eiras-Franco et al. (2019); Carrizosa et al. (2022); Bertsimas and Paskov (2022)) due to their rule-based nature.

We consider the linear regression model (the general version, with ungrouped coefficients) with n sampling units and p predictors, each of which is measured at the same T time points:

$$\begin{aligned}\mathbf{y} = &\beta_{1,1}\mathbf{x}_{1,1} + \cdots + \beta_{T,1}\mathbf{x}_{T,1} \\ &+ \beta_{1,2}\mathbf{x}_{1,2} + \cdots + \beta_{T,2}\mathbf{x}_{T,2} + \cdots \\ &+ \beta_{1,p}\mathbf{x}_{1,p} + \cdots + \beta_{T,p}\mathbf{x}_{T,p} + \boldsymbol{\epsilon},\end{aligned} \tag{1}$$

where $\mathbf{y} = \left[y_1, y_2, \ldots, y_n\right]^{\mathsf{T}}$, $\mathbf{x}_{t,j} = [x_{1,t,j}, x_{2,t,j}, \ldots, x_{n,t,j}]^{\mathsf{T}}$ for $t = 1, \ldots, T$ (our convention is that time $t = T$ corresponds to most recent, and time $t = 1$ to most distant) and $j = 1, \ldots, p$, and $\boldsymbol{\epsilon}$ are length-n vectors of the response observations, the predictor observations, and the independent and identically distributed (i.i.d.) noise with a constant variance, respectively. Here, each element of $\mathbf{x}_{t,j}$, that is, $x_{i,t,j}$, denotes the value of the jth predictor at time t for observation i. For notational simplicity, we assume predictor observations are measured at equally-spaced time points (more generally, the T time points could have unequal spacing, as long as the time points are common across all p predictors). Each $\beta_{t,j}$ is a regression coefficient for predictor j at time point t. As an example of this model, suppose we collected the data to predict political leanings (the response) based on TV viewing habits (the predictors) of n households. In this example, the index i represents a household among a total of n households, y_i is the fraction of primary voters in household i who voted in the Democrat primary, the index j represents different television programs, and so $x_{i,t,j}$ is the amount of time (e.g., 40 min) household i spent watching program j over time increment t (e.g., day t). Here, the time indices $\{1, 2, \ldots, T\}$ represent a set of T viewing time increments leading up to the election.

Depending on the structure of the response variable, different variations of (1) may be handled by our general framework. The model (1) represents the situation where $\mathbf{y}$ is not a longitudinal time series and, instead, each element of $\mathbf{y}$ represents a single response observation relevant to the entire time period $\{1, 2, \ldots, T\}$. In this situation, each row of the regression model represents a different sampling unit (e.g., a different household) at the same time. Data sets with predictors recorded over time can be considered a specific form of functional or sequential data. In addition to longitudinal predictors, the data sets with a longitudinal response that is recorded over time are often considered (see examples in Dietterich (2002); Wang et al. (2016)), although we do not consider longitudinal responses in this work. Throughout the paper, we develop the results around the situation of model (1), although the approach can also be extended to the situation in which the response $\mathbf{y}$ is observed longitudinally.

The regression model in (1) can also be written in matrix form as

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} x_{1,1,1} & \cdots & x_{1,T,1} & x_{1,1,2} & \cdots & x_{1,T,2} & \cdots & x_{1,1,p} & \cdots & x_{1,T,p} \\ x_{2,1,1} & \cdots & x_{2,T,1} & x_{2,1,2} & \cdots & x_{2,T,2} & \cdots & x_{2,1,p} & \cdots & x_{2,T,p} \\ x_{3,1,1} & \cdots & x_{3,T,1} & x_{3,1,2} & \cdots & x_{3,T,2} & \cdots & x_{3,1,p} & \cdots & x_{3,T,p} \\ \vdots & & \vdots & \vdots & & \vdots & & \vdots & \ddots & \vdots \\ x_{n,1,1} & \cdots & x_{n,T,1} & x_{n,1,2} & \cdots & x_{n,T,2} & \cdots & x_{n,1,p} & \cdots & x_{n,T,p} \end{bmatrix} \begin{bmatrix} \beta_{1,1} \\ \vdots \\ \beta_{T,1} \\ \vdots \\ \beta_{1,p} \\ \vdots \\ \beta_{T,p} \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \vdots \\ \varepsilon_n \end{bmatrix}. \tag{2}$$

Without loss of generality, we assume that each column and the response have been centered to have a sample mean zero, so that there is no intercept in the model (1).

As mentioned above, temporal dependence reflects a situation where the predictor values observed at one time point have the same (or very similar) effect on the response as do the values of predictor observations at nearby time points. Moreover, different predictors at the same time may also share the same coefficient. For example, in the television viewing example mentioned above, the times spent watching different related programs (e.g., college basketball and professional basketball; different financial news shows; etc.) will share the same coefficient if the predictor that matters most is the collective time spent watching within that group of programs; and the time spent watching basketball on a Tuesday one month prior to the primary almost certainly has a very similar effect as the time spent watching basketball on the Monday (Wednesday) immediately before (after) that particular Tuesday. In such situations, we can exploit those aspects and discover the groups of predictors that share the same coefficient according to their temporal proximity (which is partially known in advance, because the temporal ordering is known, but the cutpoints between temporal groups are not) and/or similarity of their effects on the response (which is unknown in advance and must be discovered entirely from the data). If we discover the temporal cutpoints at which each predictor within a group begins and ends, a *single derived predictor* represents the group of predictors that have the same effect on the response. To mathematically represent this single derived predictor for each group, let S_l be the set of predictor indices in group l and t_l^b and t_l^e denote the time points at which predictor observations within group l begin and end, respectively. Then, letting G_l denote the group, such that $G_l = \{S_l, [t_l^b, t_l^e]\}$ (for $l = 1, 2, \ldots, m$, where m is the number of distinct groups) and considering the derived predictor $\mathbf{z}_l = \sum_{j \in S_l} \sum_{t=t_l^b}^{t_l^e} \mathbf{x}_{t,j}$, the linear regression model in (1) can be written as

$$\mathbf{y} = \alpha_1 \mathbf{z}_1 + \alpha_2 \mathbf{z}_2 + \cdots + \alpha_m \mathbf{z}_m + \boldsymbol{\epsilon}, \tag{3}$$

where α_l represents the common coefficient shared by all predictors in G_l. Within group G_l the subsequence $(\mathbf{x}_{t_l^b,j}, \mathbf{x}_{t_l^b+1,j}, \ldots, \mathbf{x}_{t_l^e,j})$ of predictor $j \in S_l$, i.e., the observations of predictor j that are contiguous in time interval $[t_l^b, t_l^e]$, is represented by the sum $\sum_{t=t_l^b}^{t_l^e} \mathbf{x}_{t,j}$. If the unknown group structure can be discovered/identified from the data, then the subsequences of predictors that share a common regression coefficient can help users to understand the temporal patterns and the group structures, and ultimately, help users to better understand and interpret their data.

Group-structured representations in linear regression problems in which the data do not have a temporal characteristic 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 (see group lasso (Yuan and Lin 2006; Zhao et al. 2009), fused lasso (Tibshirani et al. 2005), group SCAD (Wang et al. 2007), group MCP (Breheny P 2009), group bridge (Huang et al. 2009), group hierarchical lasso (Zhou and Zhu 2010), group exponential lasso (Breheny 2015)). The second category of the

methods assumes no prior knowledge of the group structure and discovers the structure from the data (see Pelora (Dettling and Bühlmann 2004), OSCAR (Bondell and Reich 2008) and CARDS (Ke et al. 2015)). The focus of this paper is on the second category to discover predictive relationships that are not well understood in advance in addition to considering temporal characteristics from longitudinal data sets.

There are many algorithms that transform time series into simpler representations while preserving some of the underlying temporal order in the prior literature, including the discrete Fourier transformation (DFT), the discrete wavelet transformation (DWT) (Lin et al. 2003), the piecewise aggregate approximation (PAA) (Chakrabarti et al. 2002) and the symbolic aggregate approximation (SAX) (Lin et al. 2007). Those high-level representations of time series are important to reduce the dimension and extract predictors. However, the low-dimensional space is generated in an unsupervised way, without considering the effect of the predictors on the response variable, which is essential for finding our group structures.

Apart from the traditional time series representations mentioned above, tree-based time series representations have been studied in the prior literature as well. Geurts (2001) fits a regression tree using the time index t as a predictor and $x_{i,t}$, the value of the predictor at time t for observation i, as a response variable. Because the time index t is used as a predictor in this model, the terminal nodes in the tree representation represent the intervals that are contiguous in time. As another tree-based time series approach, Baydogan and Runger (2016) propose the learned pattern similarity (LPS) method based on a tree-based learning strategy. LPS extracts all possible segments of length L to construct a segment matrix using all time series observations, and then fits a regression tree by considering the random column from a segment matrix as a response variable. Baydogan et al. (2013) and Baydogan and Runger (2015) propose bag-of-words approaches to generate new predictors to be used for classification of time series. In Baydogan et al. (2013), random intervals and their characteristics such as the mean, variance, and slope are used as predictors, whereas the symbolic representation is used in Baydogan and Runger (2015). These methods extract the predictors in an unsupervised setting and then use those predictors predicting the response. In addition to these studies, there is a large body of literature on time series classification problems that extract possible subsequences (called shapelets) without considering the response when constructing the predictors, and then estimate the ones that are maximally representative of each class (Ye and Keogh 2009; Mueen et al. 2011; Ye and Keogh 2011).

One could also borrow from the usual CART (Classification and Regression Trees) framework, and modify the split-point search procedure for functional data, as was done in Balakrishnan and Madigan (2006), Belli and Vantini (2022), Möller et al. (2016), Blanquero et al. (2023). The main difference between our tree approach and the usual CART-based procedures is that our tree growing procedure splits the predictors themselves into groups to find derived predictors in a supervised manner (with each derived predictor being the sum of the predictors in each group; and for longitudinal predictors the groups are the predictor values over contiguous intervals in the longitudinal domain), whereas the CART-based tree growing procedures split the n observations into groups of observations based on the values that the predictors assume for each observation. Thus, each leaf node of our tree represents a group of predictors,

whereas each leaf node in the CART-based procedures represents a region in the predictor space. As such, our final predictive model is still a linear regression model with a parsimonious set of highly relevant derived predictors, whereas the CART-based procedures are more like standard CART predictive models. In Balakrishnan and Madigan (2006) and Belli and Vantini (2022), for example, novel splitting procedures for functional predictors are proposed within the regular CART framework, and at each functional split, the subsets of observations are split into two groups of observations. Whereas the longitudinal domain intervals in LCTR are determined in a supervised manner with the number of intervals and their widths chosen parsimoniously based on how the response depends on the longitudinal predictor, any longitudinal intervals used in forming the splitting predictors in the CART-based procedures are prespecified in an unsupervised manner without regard to the response. In parallel to this, in order to utilize the predictive accuracy of multiple trees, Möller et al. (2016) applies a random forest to the longitudinal data by generating intervals randomly drawn from an exponential distribution to construct the splitting predictors for each CART-based tree. Although combining predictions from different trees via an ensemble learner can increase the predictive accuracy, the interpretability advantage of a tree is lost due to the randomness of the intervals and the ensemble averaging. In contrast, interpretability is one of the key aspects of our approach, achieved by engineering new predictors in the form of subsequences that affect the response in a similar manner. Moreover, recently, there has been a growing interest among the optimization community to find various tree structures as an optimization problem (Blanquero et al. 2023). Similarly, one could consider attempting to find our LCTR tree structure as an optimization problem. However, solving this combinatorial optimization problem exactly is obviously computationally prohibitive. We view our proposed approach as a fast approximate solution to this large-scale optimization problem that borrows computational strategies from both linear regression and regression trees. The end result, which we demonstrate in the numerical comparison examples and which we view as the main contribution of this work, is an LCTR algorithm that provides substantially better predictive accuracy, in addition to better computational expense, relative to existing alternatives.

For non-longitudinal data, the coefficient tree regression (CTR) approach was developed to discover the unknown group structure in standard regression problems in which the data do not have a temporal characteristic (Sürer et al. 2021a, b). In this paper, we borrow concepts from Sürer et al. (2021a, b) and extend them to the longitudinal predictor case. In particular, we take advantage of the special structure of longitudinal data to develop a longitudinal CTR (LCTR) algorithm that is far more accurate than just applying the original CTR, and that still preserves the computational advantage compared to other group-based methods and fast, efficient implementation of ridge and lasso regression. LCTR is an automated way to extract the derived predictors in a supervised setting without extracting all possible subsequences.

The remainder of this paper is organized as follows. The LCTR algorithm and computational issues are discussed in Sect. 2. Sections 3 and 4 present the results of simulation studies and a real data analysis, respectively. Finally, some concluding remarks are given in Sect. 5.

2 Discovering temporal pattern in longitudinal data

Section 2.1 overviews the model building procedure for LCTR, and Sect. 2.2 introduces a toy example to illustrate the main steps of LCTR. We then describe the split-point search procedure for efficiently finding the predictor group structure in Sect. 2.3. Details regarding the model update after each iteration are given in Sect. 2.4. Section 2.5 extends LCTR for data sets comprised of a mixture of both longitudinal and non-longitudinal predictors. Finally, Sect. 2.6 describes the termination criterion for the LCTR algorithm.

2.1 Details of the model building procedure for LCTR

LCTR discovers the group structure when the groups of predictors that share the same coefficients are not known in advance. When the data have a temporal component, predictors that are closer in time are more likely to be associated with the response in a similar way. This means that predictors at nearby time points share similar coefficients, so that they should be placed into the same group. However, the predictors that belong to the group (if there are multiple predictors having similar effects, like similar programs for the example described in the introduction) and the corresponding time points at which each group begins and ends are not known in advance, and they must be discovered from the longitudinal data. The LCTR algorithm that we develop aims to discover the predictors and the corresponding temporal cutpoints at which each group begins and ends to efficiently construct derived predictors iteratively from high dimensional longitudinal data. During each iteration ($k = 1, 2, \ldots$), one of the existing groups is split into at most three groups (one split across time, and the other split across predictors, as described later and illustrated in Fig. 5) in a manner that most reduces the sum of squared errors (SSE). This is in analogy with the regular CTR algorithm, which does not consider the temporal effect and only considers a single split on the predictors at each iteration. Although the LCTR's split-point search procedure is designed to construct three subgroups from one group, it also allows subgroups to be empty depending on the best split, in which case the group is only split into two subgroups. Thus, at each iteration, an existing group is split into at most three subgroups. For the mathematical expressions throughout the paper, we assume a group is split into three subgroups, and if the number of subgroups is less than three, any empty groups are not considered for splitting in the subsequent iterations. Thus, at the end of iteration k, $2k+1$ distinct groups are constructed via LCTR, and the set $\{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{2k+1}\}$ represents the corresponding derived predictors. After each iteration k, LCTR fits a linear regression model with all derived predictors. This process continues until we obtain a final set of derived predictors satisfying a termination criterion that will be described in Sect. 2.6. In this section, we briefly explain how the groups are constructed via a toy example.

We first introduce some notation. In the context of multivariate temporal data, recall that we consider a data set of n observations and p predictors, each of which is measured at T time points. At each iteration k, the aim is to find the subset of predictors $S_{k,l}$ within group l and their temporal cutpoints $t^b_{k,l}$ and $t^e_{k,l}$ (for notational

clarity, we have added a subscript k to each group and the corresponding derived predictor and the cutpoints to denote that we obtain the model at the end of iteration k). Considering the group $G_{k,l} = \{S_{k,l}, [t^b_{k,l}, t^e_{k,l}]\}$ and the corresponding derived predictor $\mathbf{z}_{k,l} = \sum_{j \in S_{k,l}} \sum_{t=t^b_{k,l}}^{t^e_{k,l}} \mathbf{x}_{t,j}$ at the end of iteration k, the fitted model with $2k+1$ derived predictors has the following structure:

$$\begin{aligned}\hat{\mathbf{y}}(\hat{\boldsymbol{\alpha}}_k) &= \hat{\alpha}_{k,1}\mathbf{z}_{k,1} + \hat{\alpha}_{k,2}\mathbf{z}_{k,2} + \cdots + \hat{\alpha}_{k,2k+1}\mathbf{z}_{k,2k+1} \\ &= \hat{\alpha}_{k,1} \sum_{j \in S_{k,1}} \sum_{t=t^b_{k,1}}^{t^e_{k,1}} \mathbf{x}_{t,j} + \hat{\alpha}_{k,2} \sum_{j \in S_{k,2}} \sum_{t=t^b_{k,2}}^{t^e_{k,2}} \mathbf{x}_{t,j} + \cdots \\ &\quad + \hat{\alpha}_{k,2k+1} \sum_{j \in S_{k,2k+1}} \sum_{t=t^b_{k,2k+1}}^{t^e_{k,2k+1}} \mathbf{x}_{t,j},\end{aligned} \tag{4}$$

where $\hat{\boldsymbol{\alpha}}_k = \left[\hat{\alpha}_{k,1}, \hat{\alpha}_{k,2}, \ldots, \hat{\alpha}_{k,2k+1}\right]^{\mathsf{T}}$ 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 suggest that if the predictors are originally in different units, one should standardize them first or alternatively, scale them in some other meaningful way, since the group structure depends on the scaling of the predictors.

It is important to note that LCTR can obtain accurate and interpretable predictive models even when there is no explicit group structure. The LCTR approximation can be viewed as a piecewise constant approximation of the longitudinal regression coefficients. Figure 1 illustrates coefficients of $p = 3$ predictors with linear, quadratic (with a discontinuity), and cubic trends over $T = 500$ time points. Piecewise linear approximations of the coefficients using 20 pieces for each predictor are also shown in this figure (since we only have the coefficients and no data observations here, the approximations were obtained by fitting a standard regression tree to the ordered coefficients $\{\beta_{t,j} : t = 1, 2, \ldots, 500\}$ as a function of t). In all three situations, the

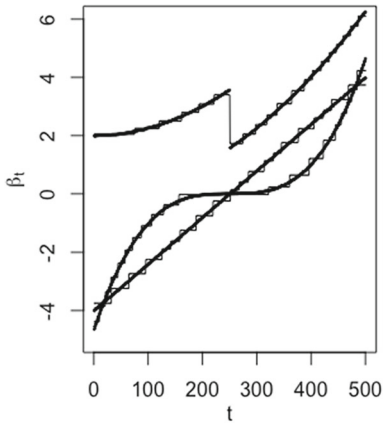


Fig. 1 Piecewise constant approximation of smoothly decaying longitudinal regression coefficients of $p = 3$ predictors with $T = 500$ time points using 20 groups for each predictor

piecewise linear functions approximate the coefficients quite closely, with training r^2 values (for representing the coefficients, not the response) very close to one. Therefore, LCTR accurately represents the true coefficients with a small number of groups even when the true coefficients do not have a group structure.

The coefficients for two of the predictors in Fig. 1 decay monotonically as one proceeds backwards in time. Although this is likely to occur in practice for many longitudinal data sets (the recent past typically influences the response more strongly than the distant past), this is not a requirement for LCTR to provide a close approximation of the true coefficients. The requirement is that the coefficients vary smoothly over a finite set of time intervals that together comprise the entire time history, with possible abrupt changes allowed between intervals. This is illustrated with the quadratic (with a discontinuity) trend in Fig. 1, which is comprised of two quadratically decaying coefficient profiles connected by a discrete jump at around $t = 250$. Most longitudinal data are likely to satisfy this requirement, except for data with strong seasonalities covering many seasonality periods. For example, if there is a weekly seasonality with large day-to-day differences and the time history covers many weeks, it would be difficult to represent the coefficients as a piecewise constant profile with a reasonably small number of pieces.

LCTR captures the temporal patterns from the data using an iterative procedure described in Algorithm 1, and the regression dependencies between different predictors and time points can be visualized with an interpretable tree structure, as illustrated in Fig. 3. In the tree, each node corresponds to a group (i.e., a set of indices of a group of predictors and the corresponding time interval that defines one derived predictor per model (4)), and the label on the branch just above the node represents the estimated coefficient of the corresponding derived predictor. The level going down into the tree from the root node corresponds to the iteration number k. At each iteration k, we call the groups with nonzero coefficients included groups, and we call the set of all predictors at all times that are not currently in any included group, which corresponds to a group with a coefficient of zero, the excluded group. At iteration $k = 0$, we start at the root node with all predictors and the time points in the excluded group, i.e., $G_{0,1} = \{\{1, \ldots, p\}, [1, T]\}$ as in line 3 of Algorithm 1, and at each iteration k we grow the tree by splitting one of the current groups into three groups. The existing groups that were not split are carried over to the next iteration without modification. Thus, the number of groups increases by two during each iteration. After iteration $k-1$, we have $2k-1$ groups. The $2k-1$ groups are disjoint groups such that if $S_{k-1,i} \cap S_{k-1,j} \neq \emptyset$ then $[t^b_{k-1,i}, t^e_{k-1,i}] \cap [t^b_{k-1,j}, t^e_{k-1,j}] = \emptyset \; \forall \{i, j\} \subseteq \{1, \ldots, 2k-1\}$ (i.e., the predictors included in two groups do not share a common time interval). During iteration k, one of the $2k-1$ groups present after iteration $k-1$ is split into three groups, and then the coefficients for all derived predictors are updated. The tree is grown until we reach some terminal number of groups based on a stopping criterion.

Algorithm 1: Longitudinal Coefficient Tree Regression

1 **Input**: $\{\mathbf{x}_{t,j} : t = 1, \ldots, T \text{ and } j = 1, \ldots, p\}$, $\mathbf{y}$, k_{max}

2 *Initialize* $(\mathbf{E}_0)_{t,j} = \sum_{t'=t}^{T} \mathbf{e}_{t',j,0} = \sum_{t'=t}^{T} \mathbf{x}_{t',j}$, $(\mathbf{w}_0)_{t,j} = \sum_{t'=t}^{T} \mathbf{e}_{t',j,0}^{\top}\mathbf{y}$, $(\mathbf{u}_0)_{t,j} = \left(\sum_{t'=t}^{T} \mathbf{e}_{t',j,0}\right)^{\top} \sum_{t'=t}^{T} \mathbf{e}_{t',j,0}$

3 *Set* $G_{0,1} = \{S_{0,1}, [t^b_{0,1}, t^e_{0,1}]\}$ where $S_{0,1} = \{1, \ldots, p\}$, $t^b_{0,1} = 1$, $t^e_{0,1} = T$

4 **for** $k = 1, \ldots, k_{max}$ **do**

5 *Initialize* $\tilde{R} = 0$

6 **for** $l = 1, \ldots, 2k-1$ **do**

7 *Split* $G_{k-1,l}$ via Algorithm 2, and obtain $\mathbf{z}_{k,l}$, $\mathbf{z}_{k,l+1}$, $\mathbf{z}_{k,l+2}$, and reduction R^* in the SSE

8 **if** $R^* > \tilde{R}$ **then**

9 $\tilde{R} = R^*$, $\tilde{k} = l$, store $\mathbf{z}_{k,\tilde{k}}$ and $\mathbf{z}_{k,\tilde{k}+1}$

10 **end**

11 **end**

12 $\mathbf{z} = \mathbf{z}_{k,\tilde{k}}$

13 $\mathbf{e}_{\mathbf{z},k-1} = \left[\mathbf{I} - \mathbf{Z}_{k-1}\left(\mathbf{Z}_{k-1}^{\mathsf{T}}\mathbf{Z}_{k-1}\right)^{-1}\mathbf{Z}_{k-1}^{\mathsf{T}}\right]\mathbf{z}$

14 $\mathbf{Z}_k = [\mathbf{Z}_{k-1}, \mathbf{z}]$, $\mathbf{v}_k = \mathbf{E}_{k-1}^{\mathsf{T}}\mathbf{e}_{\mathbf{z},k-1}$

15 $\mathbf{E}_k = \mathbf{E}_{k-1} - \frac{\mathbf{e}_{\mathbf{z},k-1}}{\mathbf{e}_{\mathbf{z},k-1}^{\mathsf{T}}\mathbf{e}_{\mathbf{z},k-1}}\mathbf{v}_k^{\mathsf{T}}$, $\mathbf{w}_k = \mathbf{w}_{k-1} - \mathbf{v}_k \frac{\mathbf{e}_{\mathbf{z},k-1}^{\mathsf{T}}\mathbf{y}}{\mathbf{e}_{\mathbf{z},k-1}^{\mathsf{T}}\mathbf{e}_{\mathbf{z},k-1}}$, $\mathbf{u}_k = \mathbf{u}_{k-1} - \frac{\mathbf{v}_k \circ \mathbf{v}_k}{\mathbf{e}_{\mathbf{z},k-1}^{\mathsf{T}}\mathbf{e}_{\mathbf{z},k-1}}$ (i.e., Equations (15)–(16))

16 If $G_{k-1,\tilde{k}}$ is a nonzero-coefficient group, set $\mathbf{z} = \mathbf{z}_{k,\tilde{k}+1}$, and $\mathbf{Z}_{k-1} = \mathbf{Z}_k$, $\mathbf{E}_{k-1} = \mathbf{E}_k$, $\mathbf{w}_{k-1} = \mathbf{w}_k$, $\mathbf{u}_{k-1} = \mathbf{u}_k$ and repeat lines 12–14

17 **end**

18 **Output**: $G_{k,l}$, $\mathbf{z}_{k,l}$, $\hat{\alpha}_{k,l}$ for $l = 1, \ldots, 2k+1$ and $k = 1, \ldots, k_{max}$

To illustrate our approach, we use a matrix to represent each group where each row and column represent the corresponding predictor and time point indices included in the group, respectively. As an example, Fig. 2 shows the $p \times T$ matrix representation of the group $G_{0,1} = \{\{1, \ldots, p\}, [1, T]\}$. During iteration k, for each group $G_{k-1,l}$ generated after iteration $k-1$ (for $l = 1, \ldots, 2k-1$ where $2k-1$ is the number of groups obtained after iteration $k-1$) we use a split-point search procedure that will be described in Sect. 2.3 to split the group into three subgroups (see line 7 of Algorithm 1). We split groups consistent with tree-structured splits of the 2D $j \times t$ (i.e., predictor-by-time) space as illustrated in Fig. 2, but after permuting the predictors

Fig. 2 The $p \times T$ matrix representation of the group $G_{0,1}$: Each row represents a predictor and each column represents a time point

$$
\begin{array}{c}
 \\ 1 \\ 2 \\ \vdots \\ j \\ \vdots \\ p
\end{array}
\begin{array}{c}
\begin{array}{cccccc} 1 & 2 & \ldots & t & \ldots & T \end{array} \\
\begin{pmatrix}
\mathbf{x}_{1,1} & \mathbf{x}_{2,1} & \ldots & \mathbf{x}_{t,1} & \ldots & \mathbf{x}_{T,1} \\
\mathbf{x}_{1,2} & \mathbf{x}_{2,2} & \ldots & \mathbf{x}_{t,2} & \ldots & \mathbf{x}_{T,2} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\mathbf{x}_{1,j} & \mathbf{x}_{2,j} & \ldots & \mathbf{x}_{t,j} & \ldots & \mathbf{x}_{T,j} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\mathbf{x}_{1,p} & \mathbf{x}_{2,p} & \ldots & \mathbf{x}_{t,p} & \ldots & \mathbf{x}_{T,p}
\end{pmatrix}
\end{array}
$$

within the group being considered for splitting (the j index) at each iteration. This has the advantage of keeping the predictors in each group contiguous in time t but allowing any grouping of predictors j. Among the $2k - 1$ possible groups to split, we choose the one that most reduces the SSE as the split group at the end of iteration k. This is implemented in lines 6–11 in Algorithm 1. Thus, we obtain $2k + 1$ distinct groups at the end of iteration k by splitting the group into three groups. This process continues until the termination criterion is satisfied as indicated in line 4 of Algorithm 1.

2.2 Illustration of interpretability via toy example

We now use the toy example in Fig. 3 to illustrate the LCTR tree growing procedure. This example is included in the vignette of `LCTR`, which is an `R` package that implements the proposed approach and that can be found in our Git repository (Sürer et al. 2022). 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. 2.3. In Fig. 3, each level corresponds to an iteration k and shows the groups produced after that iteration. This toy example uses a data set with $p = 3$, $T = 10$ and $n = 10^4$, generated based on the linear model in (1), where $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma^2\mathbf{I})$ and σ^2 was chosen to achieve a true $r^2 = 0.9$. Each regression observation (aka "row") of predictors $\left[x_{i,1,1}, x_{i,2,1}, \ldots, x_{i,10,1}, \ldots, x_{i,1,3}, x_{i,2,3}, \ldots, x_{i,10,3}\right]^{\mathsf{T}}$ is sampled from a multivariate normal distribution with mean $\mathbf{0}$ and covariance $\mathbf{I}$. The true coefficient $\beta_{t,j}$ for predictor j ($j = 1, 2, 3$) at time point t ($t = 1, 2, \ldots, 10$) is given in Fig. 4. In this example, the influence of the predictors increases with recency in time. We also included predictors with coefficients equal to zero to show how LCTR performs variable selection.

For initialization ($k = 0$), since LCTR has not identified any derived predictor yet, we have only the single group $G_{0,1} = \{\{1, 2, 3\}, [1, 10]\}$, which contains all

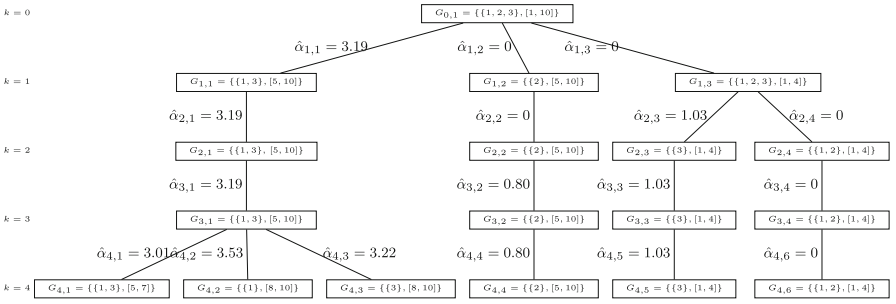


Fig. 3 Illustration of a group structure for the example in Fig. 4 produced in the LCTR tree growing procedure with $p = 3$ and $T = 10$ predictors and five nonzero- and one zero-coefficient groups in the final model

Fig. 4 The 3×10 matrix representation of the true coefficient $\beta_{j,t}$ values for $j = 1, 2, 3$, and $t = 1, \ldots, 10$

$$
\begin{array}{cc} & t \\ j & \end{array}
\begin{array}{c|cccccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
1 & 0 & 0 & 0 & 0 & 3 & 3 & 3 & 3.5 & 3.5 & 3.5 \\
2 & 0 & 0 & 0 & 0 & 0.8 & 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \\
3 & 1 & 1 & 1 & 1 & 3 & 3 & 3 & 3.2 & 3.2 & 3.2
\end{array}
$$

predictors and time points and assigns them a common coefficient of zero. During iteration $k = 1$, LCTR splits the initial group $G_{0,1}$ into three groups (see Sect. 2.3 for details of splitting procedure), where $G_{1,1} = \{\{1, 3\}, [5, 10]\}$ corresponds to the group of predictors in the fitted model, and $G_{1,2} = \{\{2\}, [5, 10]\}$ and $G_{1,3} = \{\{1, 2, 3\}, [1, 4]\}$ are the excluded groups from the model. Our splitting convention is that the groups always correspond to rectangular regions in the predictor-by-time space (i.e., the 2D (j, t) space) consistent with a tree structure after permutation of the predictors (see Fig. 5 later). After iteration $k = 1$, the model is

$$\hat{\mathbf{y}}(\hat{\boldsymbol{\alpha}}_1) = \hat{\alpha}_{1,1}\mathbf{z}_{1,1} = \hat{\alpha}_{1,1}\sum_{t=5}^{10}(\mathbf{x}_{t,1} + \mathbf{x}_{t,3}), \tag{5}$$

with $\hat{\alpha}_{1,1} = 3.19$, $\hat{\alpha}_{1,2} = 0$ and $\hat{\alpha}_{1,3} = 0$. Thus, the first derived predictor $\mathbf{z}_{1,1}$ chosen by LCTR represents the most influential group of predictors in this example.

During iteration $k = 2$, LCTR found a new group of predictors by splitting the zero-coefficient group $G_{1,3}$ into a nonzero-coefficient group $G_{2,3} = \{\{3\}, [1, 4]\}$ and a zero-coefficient group $G_{2,4} = \{\{1, 2\}, [1, 4]\}$ (in general, at each iteration we split one group into three groups, but in this case the third group is empty) while keeping the remaining groups the same, that is, $G_{2,1} = G_{1,1}$ and $G_{2,2} = G_{1,2}$. Thus, the fitted model after iteration $k = 2$ is

$$\begin{aligned}\hat{\mathbf{y}}(\hat{\boldsymbol{\alpha}}_2) &= \hat{\alpha}_{2,1}\mathbf{z}_{2,1} + \hat{\alpha}_{2,3}\mathbf{z}_{2,3} \\ &= \hat{\alpha}_{2,1}\sum_{t=5}^{10}(\mathbf{x}_{t,1} + \mathbf{x}_{t,3}) + \hat{\alpha}_{2,3}\sum_{t=1}^{4}\mathbf{x}_{t,3},\end{aligned} \tag{6}$$

with nonzero coefficients $\hat{\alpha}_{2,1} = 3.19$, $\hat{\alpha}_{2,3} = 1.03$ and zero coefficients $\hat{\alpha}_{2,2} = 0$, $\hat{\alpha}_{2,4} = 0$.

During iteration $k = 3$, the excluded group $G_{2,2} = \{\{2\}, [5, 10]\}$ becomes an included group $G_{3,2} = G_{2,2}$ having a nonzero coefficient $\hat{\alpha}_{3,2}$ (i.e., the optimal split of this group into three groups had two of the three groups empty). The remaining groups $G_{2,1}$, $G_{2,3}$ and $G_{2,4}$ were carried over to the next iteration without modification (i.e., $G_{3,1} = G_{2,1}$, $G_{3,3} = G_{2,3}$ and $G_{3,4} = G_{2,4}$). Thus, after iteration $k = 3$, the fitted model is

$$\begin{aligned}\hat{\mathbf{y}}(\hat{\boldsymbol{\alpha}}_3) &= \hat{\alpha}_{3,1}\mathbf{z}_{3,1} + \hat{\alpha}_{3,2}\mathbf{z}_{3,2} + \hat{\alpha}_{3,3}\mathbf{z}_{3,3} \\ &= \hat{\alpha}_{3,1}\sum_{t=5}^{10}(\mathbf{x}_{t,1} + \mathbf{x}_{t,3}) + \hat{\alpha}_{3,2}\sum_{t=5}^{10}\mathbf{x}_{t,2} + \hat{\alpha}_{3,3}\sum_{t=1}^{4}\mathbf{x}_{t,3},\end{aligned} \tag{7}$$

with $\hat{\alpha}_{3,1} = 3.19$, $\hat{\alpha}_{3,2} = 0.80$, $\hat{\alpha}_{3,3} = 1.03$ and $\hat{\alpha}_{3,4} = 0$. After iteration $k = 3$, all predictors that truly have nonzero coefficients have been included in the model.

Within an existing group of predictors, if some are more or less influential than others in this group, the LCTR algorithm can split them into separate subgroups at subsequent iterations to properly adjust their coefficients. For example, since the true

coefficients of predictors within group $G_{3,1}$ vary, during iteration $k = 4$, the group $G_{3,1} = \{\{1, 3\}, [5, 10]\}$ was split into three subgroups $G_{4,1} = \{\{1, 3\}, [5, 7]\}$, $G_{4,2} = \{\{1\}, [8, 10]\}$ and $G_{4,3} = \{\{3\}, [8, 10]\}$, after which the fitted model was

$$\begin{aligned}\hat{\mathbf{y}}(\hat{\boldsymbol{\alpha}}_4) &= \hat{\alpha}_{4,1}\mathbf{z}_{4,1} + \hat{\alpha}_{4,2}\mathbf{z}_{4,2} + \hat{\alpha}_{4,3}\mathbf{z}_{4,3} + \hat{\alpha}_{4,4}\mathbf{z}_{4,4} + \hat{\alpha}_{4,5}\mathbf{z}_{4,5} \\ &= \hat{\alpha}_{4,1}\sum_{t=5}^{7}(\mathbf{x}_{t,1} + \mathbf{x}_{t,3}) + \hat{\alpha}_{4,2}\sum_{t=8}^{10}\mathbf{x}_{t,1} + \hat{\alpha}_{4,3}\sum_{t=8}^{10}\mathbf{x}_{t,3} \\ &\quad + \hat{\alpha}_{4,4}\sum_{t=5}^{10}\mathbf{x}_{t,2} + \hat{\alpha}_{4,5}\sum_{t=1}^{4}\mathbf{x}_{t,3},\end{aligned} \tag{8}$$

with $\hat{\alpha}_{4,1} = 3.01$, $\hat{\alpha}_{4,2} = 3.53$, $\hat{\alpha}_{4,3} = 3.22$, $\hat{\alpha}_{4,4} = 0.80$, $\hat{\alpha}_{4,5} = 1.03$ and $\hat{\alpha}_{4,6} = 0$.

The algorithm terminated at this point, and the final fitted model includes five nonzero-coefficient groups as in (8). Thus, the number of estimated coefficients decreases from 30 (3×10) in the original regression model to five in the LCTR model. 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 $G_{4,6} = \{\{1, 2\}, [1, 4]\}$. In this sense, LCTR generates low-dimensional representation of the data via extracting derived predictors, and discards any other predictors as redundant information by including them in the zero-coefficient group.

Algorithm 2: Longitudinal Split-Point Search Procedure

Input: $G_{k-1,l} = \{S_{k-1,l}, [t^b_{k-1,l}, t^e_{k-1,l}]\}$
Initialize $R^* = 0$
for $t \in \{t^e_{k-1,l}, t^e_{k-1,l} - 1, \ldots, t^b_{k-1,l}\}$ **do**
  Construct the set of sums $\{\mathbf{s}_{t,j} = \sum_{t'=t}^{t^e_{k-1,l}} \mathbf{x}_{t',j} : j \in S_{k-1,l}\}$
  Sort the predictors in $S_{k-1,l}$ based on $\rho_{t,j}$ in Equation (11)
  Initialize $\mathbf{z}_{old} = \mathbf{z}_{new} = \mathbf{0}_{n\times 1}$, $D_{old} = D_{new} = N_{old} = N_{new} = 0$, $S^c_{k-1,l} = S_{k-1,l}$
  for $j \in S_{k-1,l}$ **do**
    Obtain $\mathbf{z}_{new} = \mathbf{z}_{old} + \mathbf{s}_{t,j}$, and set $S^c_{k-1,l} = S^c_{k-1,l} \setminus \{j\}$
    $R_{new} = N^2_{new}/D_{new}$ (i.e., Equation (10))
    Update $D_{old} = D_{new}$, $N_{old} = N_{new}$, and $\mathbf{z}_{old} = \mathbf{z}_{new}$
    if $R_{new} > R^*$ **then**
      Set $R^* = R_{new}$, $t^* = t$, $j_{t^*} = j$
      $\mathbf{z}_{k,l} = \mathbf{z}_{new}$, $\mathbf{z}_{k,l+1} = \sum_{j' \in S^c_{k-1,l}} \sum_{t'=t}^{t^e_{k-1,l}} \mathbf{x}_{t',j'}$, $\mathbf{z}_{k,l+2} = \sum_{j' \in S_{k-1,l}} \sum_{t'=t^b_{k-1,l}}^{t-1} \mathbf{x}_{t',j'}$
    end
  end
end
Output: Split point (t^*, j_{t^*}), reduction R^* in the SSE, and the derived predictors $\mathbf{z}_{k,l}$, $\mathbf{z}_{k,l+1}$, and $\mathbf{z}_{k,l+2}$

2.3 The longitudinal split-point search procedure

As mentioned above, at iteration k, for each group in $\{G_{k-1,l} : l = 1, 2, \ldots, 2k-1\}$, we use a split-point search procedure provided in Algorithm 2 to split the group and obtain the corresponding regression fit as if that group would be split after iteration k. Among $2k-1$ candidate splits, we choose the one that most reduces the SSE. Unlike when growing standard regression trees, this is equivalent to using mean squared error (MSE) to decide the best split, in the sense that it would result in the exact same sequence of splits, since MSE in our case is the SSE scaled by the total number of training observations n. LCTR and standard regression trees are different in that LCTR splits a set of predictors into subgroups of predictors, whereas regression trees split an interval of values of a single predictor into two subintervals and each region being considered for splitting contains a different number of observations. Considering all possible ways to split a given group and computing the corresponding regression fit for each possible split is computationally prohibitive, especially for large multivariate temporal data. Therefore, we develop a greedy split-point search procedure to efficiently split the group into three subgroups.

To illustrate the split-point search procedure, suppose the split group is $G_{k-1,l} = \{S_{k-1,l}, [t^b_{k-1,l}, t^e_{k-1,l}]\}$ at iteration k. To simplify the illustration, suppose $S_{k-1,l}$ includes all predictors, that is, $S_{k-1,l} = \{1, \ldots, p\}$, and the predictors are reordered and then numbered from 1 to p (if $S_{k-1,l}$ contains less than p predictors the procedure is exactly the same but with p replaced by the number of predictors in $S_{k-1,l}$) according to the sorting criterion in line 5 of Algorithm 2. The matrix in Fig. 5 represents the current group $G_{k-1,l}$, and the goal is to optimally (to most reduce the SSE) split it into three groups with the split structure shown in Fig. 5. That is, we first make a single vertical split on t and refer to the subgroup $\{\{1, 2, \ldots, p\}, [t^b_{k-1,l}, t-1]\}$ as the "left subgroup" and the remainder of $G_{k-1,l}$ as the "right subgroup". Following this, we make a single horizontal split of the right subgroup on j. For each candidate vertical split point $t \in \{t^e_{k-1,l}, t^e_{k-1,l} - 1, \ldots, t^b_{k-1,l}\}$, we find the corresponding optimal horizontal split point (denoted by j_t) by first sorting the rows of the right group and

$$
\begin{array}{c}
 \\
1 \\
2 \\
\vdots \\
j \\
j+1 \\
\vdots \\
p
\end{array}
\begin{array}{c}
\begin{array}{cccc|ccc}
t^b_{k-1,l} & t^b_{k-1,l}+1 & \ldots & t-1 & t & \ldots & t^e_{k-1,l}
\end{array} \\
\left(\begin{array}{cccc|ccc}
\mathbf{x}_{t^b_{k-1,l},1} & \mathbf{x}_{t^b_{k-1,l}+1,1} & \ldots & \mathbf{x}_{t-1,1} & \mathbf{x}_{t,1} & \ldots & \mathbf{x}_{t^e_{k-1,l},1} \\
\mathbf{x}_{t^b_{k-1,l},2} & \mathbf{x}_{t^b_{k-1,l}+1,2} & \ldots & \mathbf{x}_{t-1,2} & \mathbf{x}_{t,2} & \ldots & \mathbf{x}_{t^e_{k-1,l},2} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\mathbf{x}_{t^b_{k-1,l},j} & \mathbf{x}_{t^b_{k-1,l}+1,j} & \ldots & \mathbf{x}_{t-1,j} & \mathbf{x}_{t,j} & & \mathbf{x}_{t^e_{k-1,l},j} \\
\hline
\mathbf{x}_{t^b_{k-1,l},j+1} & \mathbf{x}_{t^b_{k-1,l}+1,j+1} & \ldots & \mathbf{x}_{t-1,j+1} & \mathbf{x}_{t,j+1} & \ldots & \mathbf{x}_{t^e_{k-1,l},j+1} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\mathbf{x}_{t^b_{k-1,l},p} & \mathbf{x}_{t^b_{k-1,l}+1,p} & \ldots & \mathbf{x}_{t-1,p} & \mathbf{x}_{t,p} & \ldots & \mathbf{x}_{t^e_{k-1,l},p}
\end{array}\right)
\end{array}
$$

Fig. 5 The goal of the split-point search is to find the split points t and j to split the group $G_{k-1,l} = \{S_{k-1,l}, [t^b_{k-1,l}, t^e_{k-1,l}]\}$ into three subgroups $G_{k,l} = \{\{1, \ldots, j\}, [t, t^e_{k-1,l}]\}$, $G_{k,l+1} = \{\{j+1, \ldots, p\}, [t, t^e_{k-1,l}]\}$ and $G_{k,l+2} = \{\{1, \ldots, p\}, [t^b_{k-1,l}, t-1]\}$ and obtain the corresponding three derived predictors $\mathbf{z}_{k,l} = \sum_{j'=1}^{j} \sum_{t'=t}^{t^e_{k-1,l}} \mathbf{x}_{t',j'}$, $\mathbf{z}_{k,l+1} = \sum_{j'=j+1}^{p} \sum_{t'=t}^{t^e_{k-1,l}} \mathbf{x}_{t',j'}$ and $\mathbf{z}_{k,l+2} = \sum_{j'=1}^{p} \sum_{t'=t^b_{k-1,l}}^{t-1} \mathbf{x}_{t',j'}$

then searching all p possible split points $j \in \{1, 2, \ldots, p\}$ for the sorted rows. In the following, we first describe the sorting procedure and then describe the horizontal (j) split point search procedure to find j_t. After finding j_t for each $t \in \{t^e_{k-1,l}, t^e_{k-1,l} - 1, \ldots, t^b_{k-1,l}\}$ and the corresponding reduction in SSE for that vertical-then-horizontal split, we choose the optimal t as the one for which the SSE was most reduced and split $G_{k-1,l}$ as in Fig. 5. This is implemented in lines 11–14 in Algorithm 2.

In the above procedure, to sort the rows of the right group for a specific $t \in \{t^e_{k-1,l}, t^e_{k-1,l} - 1, \ldots, t^b_{k-1,l}\}$, we initially obtain the sum $\mathbf{s}_{t,j} = \sum_{t'=t}^{t^e_{k-1,l}} \mathbf{x}_{t',j}$ to represent the corresponding subsequence $(\mathbf{x}_{t,j}, \mathbf{x}_{t+1,j}, \ldots, \mathbf{x}_{t^e_{k-1,l},j})$ for each predictor $j \in S_{k-1,l}$ (see line 4 of Algorithm 2). After obtaining $\mathbf{s}_{t,j}$ for each $j \in S_{k-1,l}$, as indicated in line 5, we order the predictors based on squared partial correlations between $\mathbf{s}_{t,j}$ and the response, after regressing out their dependencies on the existing derived predictors.

After sorting the rows of the right group for a specific t, we search for the optimal j_t as follows. At temporal cutpoint t, we aim to obtain the corresponding horizontal split point j_t to consider a candidate split of the group $G_{k-1,l}$. To do this, the split-point search procedure sequentially scans through the ordered elements of $S_{k-1,l}$. Using the ordered $S_{k-1,l}$ and sums $\mathbf{s}_{t,j}$ for each $j \in S_{k-1,l}$, we update the derived predictor $\mathbf{z}$ (at each t, we initially set $\mathbf{z} = \mathbf{0}_{n\times 1}$) sequentially by adding $\mathbf{s}_{t,j}$ onto the existing derived predictor $\mathbf{z}$ at each step of the split-point search procedure, and compute the corresponding reduction in the SSE as if the derived predictor $\mathbf{z}$ is included in the model. Here, each step corresponds to moving the split point down by one predictor in order as illustrated in the 2D representation in Fig. 5. Consequently, after the first step we have the derived predictor $\mathbf{z} = \mathbf{s}_{t,j_1}$ (here, j_1 represents the first predictor index in the ordered $S_{k-1,l}$), whereas $\mathbf{z} = \sum_{j\in S_{k-1,l}} \mathbf{s}_{t,j}$ after the last step at time t. Among $|S_{k-1,l}|$ different horizontal split points, j_t is the one at which the SSE is minimized. We repeat this procedure for each $t \in \{t^e_{k-1,l}, t^e_{k-1,l} - 1, \ldots, t^b_{k-1,l}\}$, and among all (t, j_t) pairs, the one that most improves the fit is selected as the best split point. Consequently, to split the group $G_{k-1,l}$, we consider $(t^e_{k-1,l} - t^b_{k-1,l} + 1) \times |S_{k-1,l}|$ possible splits, and then we pick the one that most reduces the SSE (see lines 3–16 of Algorithm 2).

Returning to our toy example in Fig. 3, at iteration $k = 1$, the excluded group $G_{0,1} = \{\{1, 2, 3\}, [1, 10]\}$ is split into three subgroups $G_{1,1} = \{\{1, 3\}, [5, 10]\}$, $G_{1,2} = \{\{2\}, [5, 10]\}$ and $G_{1,3} = \{\{1, 2, 3\}, [1, 4]\}$ as illustrated in Fig. 6. In this example, $t^b_{0,1} = 1$ and $t^e_{0,1} = 10$, and $|S_{0,1}| = 3$. Therefore, there are $30 = (10 - 1 + 1) \times 3$ candidate splits, and among those candidates we choose the one whose split most improves the fit. Towards this end, we first check the candidate vertical split point $t = 10$ and find

$$
\begin{array}{c}
\\
1 \\
3 \\
\\
2
\end{array}
\begin{array}{c}
\begin{array}{cccccc} 1 & \ldots & 4 & 5 & \ldots & 10 \end{array} \\
\left(\begin{array}{cccccc}
\mathbf{x}_{1,1} & \ldots & \mathbf{x}_{4,1} & \mathbf{x}_{5,1} & \ldots & \mathbf{x}_{10,1} \\
\mathbf{x}_{1,3} & \ldots & \mathbf{x}_{4,3} & \mathbf{x}_{5,3} & \ldots & \mathbf{x}_{10,3} \\
\hline
\mathbf{x}_{1,2} & \ldots & \mathbf{x}_{4,2} & \mathbf{x}_{5,2} & \ldots & \mathbf{x}_{10,2}
\end{array}\right)
\end{array}
$$

Fig. 6 For the Fig. 3 example, illustration of the splits at iteration $k = 1$. After iteration $k = 1$, the zero-coefficient group $G_{0,1} = \{\{1, 2, 3\}, [1, 10]\}$ is split into three groups $G_{1,1} = \{\{1, 3\}, [5, 10]\}$, $G_{1,2} = \{\{2\}, [5, 10]\}$ and $G_{1,3} = \{\{1, 2, 3\}, [1, 4]\}$

its corresponding best horizontal split point j_{10} as follows. We obtain $\mathbf{s}_{10,1} = \mathbf{x}_{10,1}$, $\mathbf{s}_{10,2} = \mathbf{x}_{10,2}$ and $\mathbf{s}_{10,3} = \mathbf{x}_{10,3}$ for each predictor in $S_{0,1} = \{1, 2, 3\}$, respectively, and order the predictors based on partial correlations between $\mathbf{s}_{10,1}$, $\mathbf{s}_{10,2}$, $\mathbf{s}_{10,3}$ and the response after regressing out their dependencies on the existing derived predictors (in this case, at iteration $k = 0$ there is no derived predictor in the model as initialization). We then obtain the ordered $S_{0,1} = \{3, 1, 2\}$. As illustrated in Fig. 7, because the third predictor is the first predictor in the ordered $S_{0,1} = \{3, 1, 2\}$, we initially obtain $\mathbf{z} = \mathbf{s}_{10,3}$. As a next step, we move the split point down by one predictor in order and update $\mathbf{z} = \mathbf{s}_{10,3} + \mathbf{s}_{10,1}$. Finally, we have $\mathbf{z} = \mathbf{s}_{10,3} + \mathbf{s}_{10,1} + \mathbf{s}_{10,2}$. For each candidate derived predictor $\mathbf{z}$, we compute the reduction in the SSE as if $\mathbf{z}$ is included in the model (the details to compute the SSE reduction will be explained later in this section). It turned out that $j_{10} = 1$ is the optimal horizontal split point at temporal cutpoint $t = 10$ at which the reduction in the SSE is maximized. We repeat this procedure for each $t \in \{10, 9, \ldots, 1\}$, and at each time t find the corresponding best horizontal split. Finally, we select the split points t and j_t as the ones that most reduce the SSE.

We now explain how to compute the reduction in the SSE when adding a new derived predictor $\mathbf{z}$ (for any $\mathbf{z}$) into the model in the general case. Without loss of generality, suppose that the groups are reordered so that the first $\tilde{k}$ groups are the included groups. Thus, $\mathbf{Z}_{k-1} = \left[\mathbf{z}_{k-1,1}, \mathbf{z}_{k-1,2}, \ldots, \mathbf{z}_{k-1,\tilde{k}}\right]$ represents all nonzero derived predictors in the fitted model at the end of iteration $k-1$. In order to compute the reduction in SSE when the derived predictors grow from $\mathbf{Z}_{k-1}$ to $\left[\mathbf{Z}_{k-1}, \mathbf{z}\right]$ in a computationally efficient manner, we use well-known geometric arguments for least squares with orthogonal projections as shown in Sürer et al. (2021a). Consequently,

$$
\begin{array}{c} \\ 3 \\ 1 \\ 2 \end{array}
\begin{array}{c}
\begin{array}{cccccc} 1 & 2 & \ldots & 9 & 10 & \end{array} \\
\begin{pmatrix}
\mathbf{x}_{1,3} & \mathbf{x}_{2,3} & \ldots & \mathbf{x}_{9,3} & \mathbf{x}_{10,3} \\
\mathbf{x}_{1,1} & \mathbf{x}_{2,1} & \ldots & \mathbf{x}_{9,3} & \mathbf{x}_{10,1} \\
\mathbf{x}_{1,2} & \mathbf{x}_{2,2} & \ldots & \mathbf{x}_{9,3} & \mathbf{x}_{10,2}
\end{pmatrix}
\end{array}
\begin{array}{c} \\ \mathbf{s}_{10,3} \\ \\ \end{array}
$$

(a) Obtain $\mathbf{z} = \mathbf{s}_{10,3}$ after the first step and compute the SSE

$$
\begin{array}{c} \\ 3 \\ 1 \\ 2 \end{array}
\begin{array}{c}
\begin{array}{cccccc} 1 & 2 & \ldots & 9 & 10 & \end{array} \\
\begin{pmatrix}
\mathbf{x}_{1,3} & \mathbf{x}_{2,3} & \ldots & \mathbf{x}_{9,3} & \mathbf{x}_{10,3} \\
\mathbf{x}_{1,1} & \mathbf{x}_{2,1} & \ldots & \mathbf{x}_{9,3} & \mathbf{x}_{10,1} \\
\mathbf{x}_{1,2} & \mathbf{x}_{2,2} & \ldots & \mathbf{x}_{9,3} & \mathbf{x}_{10,2}
\end{pmatrix}
\end{array}
\begin{array}{c} \\ \mathbf{s}_{10,3} \\ \mathbf{s}_{10,1} \\ \end{array}
$$

(b) Obtain $\mathbf{z} = \mathbf{s}_{10,3} + \mathbf{s}_{10,1}$ after the second step and compute the SSE

$$
\begin{array}{c} \\ 3 \\ 1 \\ 2 \end{array}
\begin{array}{c}
\begin{array}{cccccc} 1 & 2 & \ldots & 9 & 10 & \end{array} \\
\begin{pmatrix}
\mathbf{x}_{1,3} & \mathbf{x}_{2,3} & \ldots & \mathbf{x}_{9,3} & \mathbf{x}_{10,3} \\
\mathbf{x}_{1,1} & \mathbf{x}_{2,1} & \ldots & \mathbf{x}_{9,3} & \mathbf{x}_{10,1} \\
\mathbf{x}_{1,2} & \mathbf{x}_{2,2} & \ldots & \mathbf{x}_{9,3} & \mathbf{x}_{10,2}
\end{pmatrix}
\end{array}
\begin{array}{c} \\ \mathbf{s}_{10,3} \\ \mathbf{s}_{10,1} \\ \mathbf{s}_{10,2} \end{array}
$$

(c) Obtain $\mathbf{z} = \mathbf{s}_{10,3} + \mathbf{s}_{10,1} + \mathbf{s}_{10,2}$ after the third step and compute the SSE

Fig. 7 For the Fig. 6 example, at the candidate vertical split point $t = 10$, illustration of finding the horizontal split point j_{10} after obtaining ordered $S_{0,1} = \{3, 1, 2\}$

letting $R(\mathbf{z})$ denote the reduction in SSE with the addition of $\mathbf{z}$ into the model, we have

$$R(\mathbf{z}) = \frac{(\mathbf{e}_\mathbf{y}^\mathsf{T}\mathbf{e}_\mathbf{z})^2}{\mathbf{e}_\mathbf{z}^\mathsf{T}\mathbf{e}_\mathbf{z}} = \frac{N^2}{D}, \tag{9}$$

where $N \equiv \mathbf{e}_\mathbf{z}^\mathsf{T}\mathbf{e}_\mathbf{y}$ and $D \equiv \mathbf{e}_\mathbf{z}^\mathsf{T}\mathbf{e}_\mathbf{z}$. Here, $\mathbf{e}_\mathbf{z} = \left[\mathbf{I} - \mathbf{Z}_{k-1}(\mathbf{Z}_{k-1}^\mathsf{T}\mathbf{Z}_{k-1})^{-1}\mathbf{Z}_{k-1}^\mathsf{T}\right]\mathbf{z}$ and $\mathbf{e}_\mathbf{y} = \left[\mathbf{I} - \mathbf{Z}_{k-1}(\mathbf{Z}_{k-1}^\mathsf{T}\mathbf{Z}_{k-1})^{-1}\mathbf{Z}_{k-1}^\mathsf{T}\right]\mathbf{y}$ are the errors in projecting $\mathbf{z}$ and $\mathbf{y}$ onto the span of $\mathbf{Z}_{k-1}$, respectively. Now, suppose that during the search procedure at cutpoint t we ordered the predictors (i.e., rows in 2D representation as in Fig. 5) as explained above, and obtained the reduction $R(\mathbf{z}_{old})$ in the SSE for a derived predictor $\mathbf{z}_{old}$ such that $R(\mathbf{z}_{old}) = \frac{N_{old}^2}{D_{old}}$. Then, for the next predictor in the order (say predictor with index j), we need to compute the reduction $R(\mathbf{z}_{new})$ in the SSE, where $\mathbf{z}_{new} = \mathbf{z}_{old} + \mathbf{s}_{t,j} = \mathbf{z}_{old} + \sum_{t'=t}^{t_{k-1,l}^e} \mathbf{x}_{t',j}$. Instead of computing $R(\mathbf{z}_{new})$ via (9), one can show that $R(\mathbf{z}_{new})$ can be computed efficiently using the numerator and denominator of $R(\mathbf{z}_{old})$ as

$$R(\mathbf{z}_{new}) = \frac{\left(N_{old} + \mathbf{e}_\mathbf{y}^\mathsf{T}\sum_{t'=t}^{t_{k-1,l}^e}\mathbf{e}_{t',j,k-1}\right)^2}{D_{old} + 2\mathbf{e}_{\mathbf{z}_{old}}^\mathsf{T}\sum_{t'=t}^{t_{k-1,l}^e}\mathbf{e}_{t',j,k-1} + \left(\sum_{t'=t}^{t_{k-1,l}^e}\mathbf{e}_{t',j,k-1}\right)^\mathsf{T}\left(\sum_{t'=t}^{t_{k-1,l}^e}\mathbf{e}_{t',j,k-1}\right)}, \tag{10}$$

where $\mathbf{e}_{t,j,k-1} = \mathbf{x}_{t,j} - \mathbf{P}_{k-1}\mathbf{x}_{t,j}$ and $\mathbf{P}_{k-1} = \left[\mathbf{Z}_{k-1}(\mathbf{Z}_{k-1}^\mathsf{T}\mathbf{Z}_{k-1})^{-1}\mathbf{Z}_{k-1}^\mathsf{T}\right]$.

In the split-point search procedure updates, the decision on how to split the predictors into groups is based on the squared partial correlations $\rho_{t,j}$ between $\mathbf{s}_{t,j}$ and the response $\mathbf{y}$, after regressing out their dependencies on the existing derived predictors, where

$$\rho_{t,j} = \frac{\left(\mathbf{e}_\mathbf{y}^\mathsf{T}\sum_{t'=t}^{t_{k-1,l}^e}\mathbf{e}_{t',j,k-1}\right)^2}{\left(\sum_{t'=t}^{t_{k-1,l}^e}\mathbf{e}_{t',j,k-1}\right)^\mathsf{T}\left(\sum_{t'=t}^{t_{k-1,l}^e}\mathbf{e}_{t',j,k-1}\right)}, \tag{11}$$

and it is the marginal reduction in SSE for each sum $\mathbf{s}_{t,j}$ as if the derived predictors grow from $\mathbf{Z}_{k-1}$ to $[\mathbf{Z}_{k-1}, \mathbf{s}_{t,j}]$. Theorem 1 in Appendix 1 provides a justification for our ordering procedure. According to Theorem 1, the rationale of the ordering procedure is that sums with the largest values with squared partial correlation are good candidates for reducing the SSE. When ordering predictors in $S_{k-1,l}$, in addition to marginal reduction in SSE for each sum, we consider the sign of the correlation coefficient. If the largest squared partial correlation has a negative correlation coefficient (i.e., $\mathbf{e}_\mathbf{y}^\mathsf{T}\sum_{t'=t}^{t_{k-1,l}^e}\mathbf{e}_{t',j,k-1}$ is negative), we sort the rows in ascending order, otherwise we sort them in descending order. If we only consider the marginal SSE reduction, then the two sums with partial correlations different in sign can be included in the same group.

In this case, the negative and positive values (say, for N_{old} and $\mathbf{e}_{\mathbf{y}}^{\mathsf{T}} \sum_{t'=t}^{t^e_{k-1,l}} \mathbf{e}_{t',j,k-1}$, respectively) in the numerator of (10) would cancel each other resulting in the low SSE reduction even though their marginal SSE reduction is large. In this sense, our sorting criterion encourages the sums of predictors that affect the response in the same direction to be included in the same group at iteration k.

2.4 Updating the model at the end of each iteration

The split-point search procedure is the same for both zero- and nonzero-coefficient groups. However, the way the model is updated after the split slightly differs depending on whether the split group is a zero- or nonzero-coefficient group, which we describe in the following.

First, suppose that one of the zero-coefficient groups (i.e., one of the excluded groups) is split during iteration k and say the split group is $G_{k-1,2k-1}$ to illustrate the model update. As mentioned in Sect. 2.3, we construct groups $G_{k,2k-1}$, $G_{k,2k}$ and $G_{k,2k+1}$ and the corresponding derived predictors $\mathbf{z}_{k,2k-1}$, $\mathbf{z}_{k,2k}$ and $\mathbf{z}_{k,2k+1}$ such that $\mathbf{z}_{k,2k-1}$ is the only derived predictor with a nonzero-coefficient. Then, the nonzero-coefficient derived predictors after iteration k are $\mathbf{Z}_k = \left[\mathbf{Z}_{k-1}, \mathbf{z}\right]$ for a single new derived predictor $\mathbf{z} = \mathbf{z}_{k,2k-1}$. Consequently, at the end of iteration k, splitting a zero-coefficient group into three subgroups increases the number of derived predictors in the model by only one.

For our example shown in Fig. 3, recall that at iteration $k = 1$ the zero-coefficient group $G_{0,1}$ is split into three groups $G_{1,1}$, $G_{1,2}$ and $G_{1,3}$ as in Fig. 6. The coefficients of the corresponding derived predictors resulting from the split of zero-coefficient group $G_{0,1}$ are $\hat{\alpha}_{1,1} = 3.19$, $\hat{\alpha}_{1,2} = 0$ and $\hat{\alpha}_{1,3} = 0$ as shown in Fig. 3. As a result, the number of the derived predictors in the model increases by one (in this case, the number of derived predictors increases to 1 from 0 at iteration $k = 1$).

Suppose instead that one of the included groups in $\{G_{k-1,l} : l = 1, 2, \ldots, \tilde{k}\}$ is split during iteration k and (without loss of generality) the groups are reordered so that the split group is $G_{k-1,\tilde{k}}$. On the surface, splitting one of the existing groups actually creates three new derived predictors and removes one existing derived predictor in the model. In this case, the $\tilde{k} - 1$ predictors $\left[\mathbf{z}_{k-1,1}, \mathbf{z}_{k-1,2}, \ldots, \mathbf{z}_{k-1,\tilde{k}-1}\right]$ remain unchanged in the model during iteration k, so that the derived predictors after iteration k are $\mathbf{Z}_k = \left[\mathbf{z}_{k-1,1}, \mathbf{z}_{k-1,2}, \ldots, \mathbf{z}_{k-1,\tilde{k}-2}, \mathbf{z}_{k-1,\tilde{k}-1}, \mathbf{z}_{k,\tilde{k}}, \mathbf{z}_{k,\tilde{k}+1}, \mathbf{z}_{k,\tilde{k}+2}\right]$, whereas the derived predictors after iteration $k - 1$ are $\mathbf{Z}_{k-1} = \left[\mathbf{z}_{k-1,1}, \mathbf{z}_{k-1,2}, \ldots, \mathbf{z}_{k-1,\tilde{k}}\right]$. However, the four sets $\left[\mathbf{z}_{k,\tilde{k}}, \mathbf{z}_{k,\tilde{k}+1}, \mathbf{z}_{k,\tilde{k}+2}\right]$, $\left[\mathbf{z}_{k-1,\tilde{k}}, \mathbf{z}_{k,\tilde{k}+1}, \mathbf{z}_{k,\tilde{k}+2}\right]$, $\left[\mathbf{z}_{k-1,\tilde{k}}, \mathbf{z}_{k,\tilde{k}}, \mathbf{z}_{k,\tilde{k}+2}\right]$ and $\left[\mathbf{z}_{k-1,\tilde{k}}, \mathbf{z}_{k,\tilde{k}}, \mathbf{z}_{k,\tilde{k}+1}\right]$ all span the same three-dimensional subspace, because $\mathbf{z}_{k,\tilde{k}}$, $\mathbf{z}_{k,\tilde{k}+1}$ and $\mathbf{z}_{k,\tilde{k}+2}$ were formed by a single split of $\mathbf{z}_{k-1,\tilde{k}}$. Thus, $\mathbf{Z}_k$ and $\left[\mathbf{Z}_{k-1}, \mathbf{z}_{k,\tilde{k}}, \mathbf{z}_{k,\tilde{k}+1}\right]$ (or $\left[\mathbf{Z}_{k-1}, \mathbf{z}_{k,\tilde{k}}, \mathbf{z}_{k,\tilde{k}+2}\right]$ or $\left[\mathbf{Z}_{k-1}, \mathbf{z}_{k,\tilde{k}+1}, \mathbf{z}_{k,\tilde{k}+2}\right]$) span the same $(\tilde{k} + 2)$-dimensional subspace. In such a case, after iteration k the dimension of the model increases by only two, and we can update the model via a rank-2 update, which is equivalent to updating the model via two rank-1 updates (that is, first update the model from $\left[\mathbf{Z}_{k-1}\right]$ to $\left[\mathbf{Z}_{k-1}, \mathbf{z}_{k,\tilde{k}}\right]$, and then update it from $\left[\mathbf{Z}_{k-1}, \mathbf{z}_{k,\tilde{k}}\right]$ to $\left[\mathbf{Z}_{k-1}, \mathbf{z}_{k,\tilde{k}}, \mathbf{z}_{k,\tilde{k}+1}\right]$). Efficient rank-1 updates are provided in Appendix 2.

Returning to our toy example to illustrate the procedure, at iteration $k = 4$, the nonzero-coefficient group $G_{3,1}$ is split into three subgroups, and the estimated coefficients of the subgroups resulting from the split are $\hat{\alpha}_{4,1} = 3.01$, $\hat{\alpha}_{4,2} = 3.53$ and $\hat{\alpha}_{4,3} = 3.22$. In such a case, on the surface (as illustrated in the tree structure as well), we removed the derived predictor corresponding to $G_{3,1}$ from the model, and created three new derived predictors corresponding to $G_{4,1}$, $G_{4,2}$ and $G_{4,3}$. The remaining groups stayed the same, i.e., $G_{4,4} = G_{3,2}$, $G_{4,5} = G_{3,3}$ and $G_{4,6} = G_{3,4}$. However, in our computations we used a rank-1 update twice to efficiently update the model.

2.5 Including non-longitudinal predictors in LCTR

LCTR is designed to discover temporal pattern from the data. However, in many settings, data sets include both longitudinal and non-longitudinal predictors at the same time. For example, in the television viewing example mentioned in the introduction, in addition to TV viewing habits over time, the average annual salary and/or the education level of an household are two non-longitudinal predictors that can help predicting the political leanings of each household in the data set. In such situations, LCTR can be modified to discover the group structure existing in both the longitudinal and non-longitudinal predictors.

As described above, LCTR starts with a single group with all predictor observations included in group $G_{0,1} = \{\{1, \ldots, p\}, [1, T]\}$. When there exists predictors that do not have a longitudinal characteristic (i.e., static) as well, for the sake of completeness, we order them and label as predictors $\tilde{p} + 1, \ldots, p$, where $\tilde{p}$ is the number of longitudinal predictors, and $p - \tilde{p}$ is the number of non-longitudinal predictors. In this case, at iteration $k = 0$, one can create an additional zero-coefficient group of all non-longitudinal predictors, which we denote by $G^{nl}_{0,1} = \{\tilde{p} + 1, \ldots, p\}$. Then, at each iteration, in addition to the groups of longitudinal predictors, one can consider splitting groups of non-longitudinal predictors, and choosing the split that most improves the SSE. Each group of non-longitudinal predictors can be represented in a one-dimensional predictor space, and the split-point search procedure can be implemented as if the group is a longitudinal group but with the ending time point equal to the beginning time point, in which case no temporal split search is conducted. Non-longitudinal predictors are therefore only split on predictors and not on time.

2.6 Finding the size of a longitudinal coefficient tree regression

LCTR grows the tree structure by splitting a group into three subgroups and constructing the corresponding derived predictors at each iteration. Thus, at the end of iteration k, LCTR constructs $2k + 1$ distinct groups, and we have $2k + 1$ nodes in the corresponding tree structure as represented in Fig. 3. We need a termination criterion to determine the best k number of iterations. The total number of iterations at termination is the only tuning parameter for LCTR, and we use cross validation (CV) to find the best parameter value.

We use 10-fold CV in our experiments. Care must be taken to do every step of the model building within CV to avoid overfitting. This is analogous to how searching

and sequentially adding predictors in stepwise regression must be done within CV, since doing it as a pre-processing step prior to CV will result in overfitting. We first divide the data set into 10 equally-sized folds. For the ith hold-out fold, we fit the LCTR model using the remaining 9 folds as a training set with some sufficiently large number of iterations k_{max}. Then, for each model with the number of iterations ranging from 1 to k_{max}, we obtain the CV SSE to predict the ith fold as the validation set. We repeat this procedure 10 times, each time leaving out a different fold for the validation set and taking the remaining 9 folds as the training set. Finally, we average the CV SSE across 10 folds to give the overall CV SSE for each of the k_{max} models, and choose the best k as the one that minimizes the overall CV SSE. To obtain the final group structure, we fit the LCTR model with the number of iterations set equal to the best k.

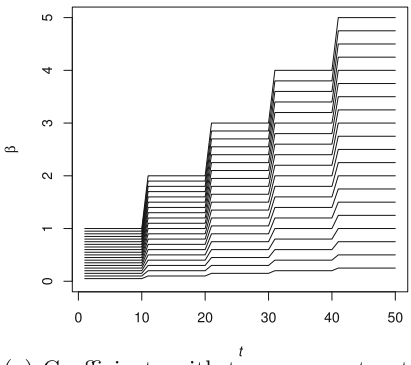
The user must select a suitable k_{max} value to obtain the best k. To accomplish this, we recommend initially using some moderate k_{max} value depending on a user's choice. If the best k obtained through CV is equal to this initial k_{max}, this is an indication that the best k is larger than the initial k_{max} value. In this case, a user should increase the k_{max} value to a larger value. If this is the case, instead of fitting each CV model from scratch, one can use a warm-start strategy by inputting the outputs (including the fitted CV models with number of iterations ranging from 1 to the initial k_{max}) for the initial k_{max} value to the new model. This warm-start feature is implemented in our `LCTR` R package.

3 Experiments

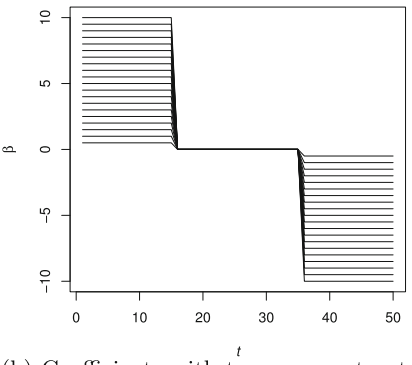
This section compares the performance of LCTR with various methods under different settings. Section 3.1 provides the details of the experiment designs used in our numerical experiments. Section 3.2 presents the main takeaways from the experiment results.

3.1 Settings

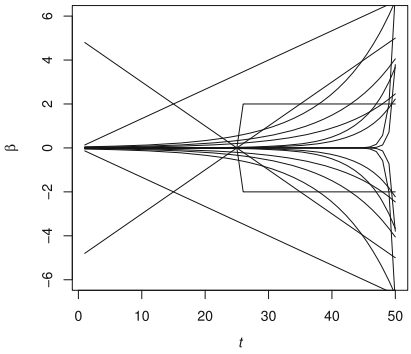
We first examine the predictive and computational performance of LCTR in comparison with ridge and lasso. We also include OLS to serve as reference for comparison. We use the `R` language package `glmnet` for ridge and lasso (Friedman et al. 2010) and `lm` for OLS (Team 2017). For LCTR, we implement our own `R` package `LCTR`. We then provide examples to illustrate the advantages of LCTR, relative to some well-known unsupervised models based on regression trees and the piecewise aggregate approximation (PAA) and the original CTR, which does not incorporate any longitudinal characteristics. Regression trees and PAA are generated through the `rpart` (Therneau and Atkinson 2019) and `TSrepr` (Laurinec 2018) packages, and CTR is built with `CTR` R package (Sürer et al. 2021c). Finally, LCTR is tested with a real data set that includes both longitudinal and non-longitudinal predictors. All experiments were completed using hardware consisting of a computer with macOS Sierra operating system and 2.5 GHz processor with 16 GB RAM.



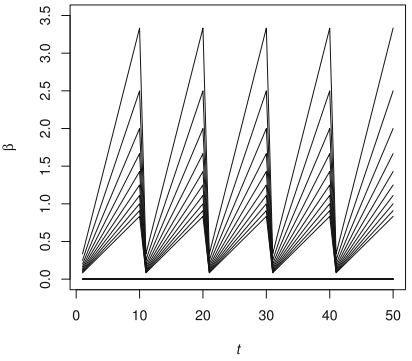
(a) Coefficients with true group structure without sparsity



(b) Coefficients with true group structure with sparsity



(c) Coefficients with smoothly decaying curves

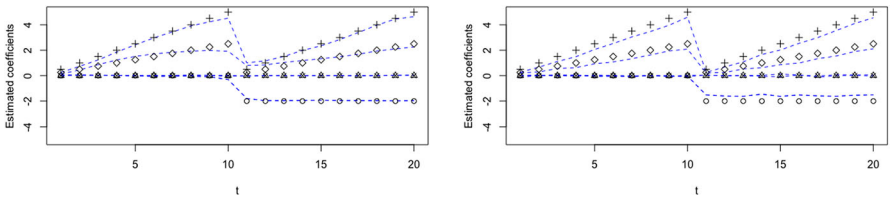


(d) Coefficients with seasonality and sparsity (half of the predictor having zero coefficients)

Fig. 8 True coefficient distributions for four different experiments with $p = 20$ and $T = 50$

Our initial simulations compared the accuracy, variable selection performance, and run times of different models on data generated from the regression model in (1), where $\boldsymbol{\epsilon} \sim \mathrm{N}(\mathbf{0}, \sigma^2\mathbf{I})$ and each row of $[\mathbf{x}_{1,1}, \ldots, \mathbf{x}_{T,1}, \ldots, \mathbf{x}_{1,p}, \ldots, \mathbf{x}_{T,p}]$ is $\mathrm{MVN}_{p\times T}(\mathbf{0}, \mathbf{I})$. We used $p = 20$ and $T = 50$ so that in total we have 1000 predictors and varied the number of observations as $n \in \{1500, 2000, 3000\}$ for the accuracy comparisons. Smaller data sets are used to evaluate the variable selection performance and larger data sets are used for the run-time comparisons described later. We considered a number of different true models, each with a different $\boldsymbol{\beta} = [\beta_{1,1}, \ldots, \beta_{T,1}, \ldots, \beta_{1,p}, \ldots, \beta_{T,p}]$, and noise level. For all experiments, we 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 with independent predictors, we organize the true $\boldsymbol{\beta}$ in four categories as in Fig. 8. Each line in the plot shows the coefficient values for a predictor's observations over time. Figure 8a–b are used to illustrate the predictive



(a) LCTR (b) Lasso

Fig. 9 Estimated coefficients for the experiment with $p = 5$, $T = 20$, $r^2 = 0.7$, and $n = 300$. Markers and dashed line illustrate the true and estimated coefficients, respectively

performance of LCTR when there is a true group structure without and with sparsity, respectively. In Fig. 8a, the coefficients of all predictors are positive and decay to zero over time, whereas in Fig. 8b the coefficients vary in magnitude and flip signs over time. Moreover, our simulations compared the accuracy when there is no actual group structure as in Fig. 8c–d. In Fig. 8c, the effect of coefficients smoothly changes over time, whereas in Fig. 8d, the coefficient values show strongly seasonal effects. These last two examples were included to demonstrate that LCTR still performs quite well even when there is no actual group structure in the predictors. We then investigate how LCTR's split-point search procedure allows variable selection across both time and predictor space. We use a simulation experiment with a true $\boldsymbol{\beta}$ given in Fig. 9 with $p = 5$, $T = 20$ and $n \in \{150, 200, 300\}$. In Fig. 9, markers indicate the true coefficients for each predictor. Thus, there are two predictors with entirely nonzero coefficients, two with entirely zero coefficients, and one with a mixture of zero and nonzero coefficients. In total, we have 50 nonzero- and 50 zero-coefficient predictors. We also considered larger data sets when investigating the computational performance of LCTR. For these experiments, we used the setting in Fig. 8d with $n \in \{10^3, 10^4, 10^5, 10^6\}$ and $p \in \{10, 20, 40, 80\}$ and $T = 50$.

Our approach assumes the errors ε are i.i.d., but autocorrelation in the predictors is allowed, so that any temporal autocorrelation in the response is due to its dependence on the predictors. In addition to synthetic data sets introduce above, we consider examples with autocorrelated predictors and certain interpretability implications. Dividing longitudinal data into subsequences and replacing the predictor observations with their sums (or their means) have certain interpretability advantages since it preserves the underlying temporal characteristics and reduces the dimension by creating new predictors in the form of sums. Moreover, representing many predictors with their sum reduces the noise, and thus helps improve the predictive accuracy. However, if the subsequences are constructed without considering the predictors' effects on the response variable, the derived predictors are not necessarily relevant to predicting the response variable, and thus the predictive accuracy decreases. The goal of LCTR is to identify the critical subsequences in a supervised way to enhance both interpretability and the predictive accuracy of a linear regression model.

To illustrate this, we first compare LCTR with PAA method (Chakrabarti et al. 2002) using a synthetic longitudinal data set generated via random walks with a single predictor (i.e., $p = 1$) observed over time $T = 60$. Among other transformation

methods such as DFT and DWT, we use PAA to find derived predictors in an unsupervised manner, since it has predictors that would be most similar to the derived predictors of LCTR which seem the most relevant for these examples. We considered a true model where $\beta_{i,1} = 0$ for $i = 1, \ldots, 27$, $\beta_{i,1} = -3$ for $i = 28, \ldots, 49$, $\beta_{i,1} = 3$ for $i = 50, \ldots, 57$, and $\beta_{i,1} = 10$ for $i = 58, \ldots, 60$. Thus, there are three nonzero coefficient groups and a zero-coefficient group, and the influence of predictor observations increases with recency in time. We varied the number of observations as $n \in \{60, 90, 120\}$.

To divide the longitudinal data into different-length subsequences, one could also use traditional regression trees by considering time index t as a predictor and the value of a predictor at time t as the response variable (Geurts 2001; Baydogan and Runger 2016). Once a tree is constructed, the split points across time are used to construct the derived predictors. To illustrate this, we used the same true coefficient structure as in the example above, and each row of $[\mathbf{x}_{1,1}, \ldots, \mathbf{x}_{T,1}]$ is $\text{MVN}_T(\mathbf{0}, \mathbf{6})$ where $\mathbf{6}_{t,t'} = 0.7^{|t-t'|}$ for $t, t' = 1, \ldots, T$.

Unlike the existing methods discussed above, CTR (Sürer et al. 2021a) was developed to discover the group structure in a supervised way. CTR finds 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 SSE. However, when finding splits, CTR does not consider the temporal characteristic of predictor observations at T time points, and only considers splitting on the predictors at each iteration. In other words, for a longitudinal data set with p predictors and T time points, instead of having a 2D predictor-time space over which to search, as illustrated in Fig. 2, CTR only considers a one-dimensional predictor space with a total of $p \times T$ predictors.

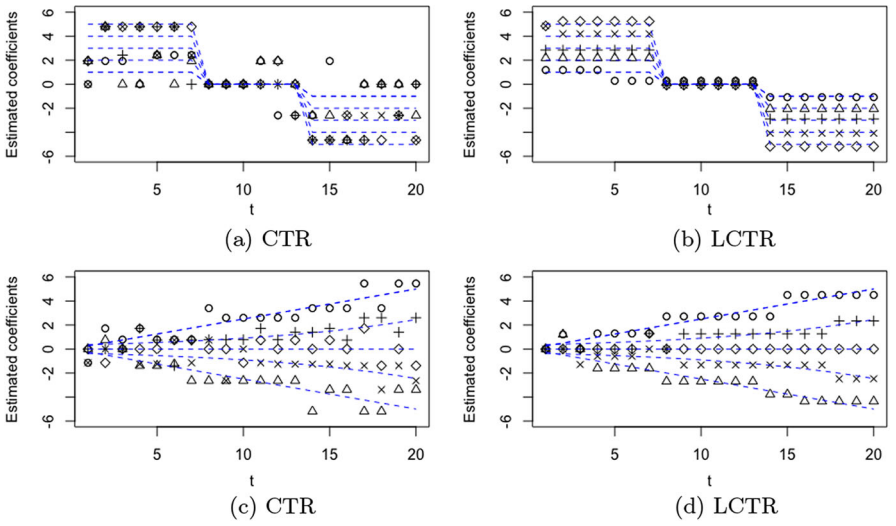


Fig. 10 Estimated coefficients for the experiment with $p = 5$, $T = 20$, and $r^2 = 0.9$. Blue dashed lines show the true coefficients of five predictors over time $T = 20$, and each marker illustrates the estimated coefficients for each predictor

To compare CTR and LCTR, we considered two true models for $\boldsymbol{\beta}$ with $p = 5$ and $T = 20$, each of which is illustrated in Fig. 10 as blue dashed lines. Each observation is sampled from a multivariate normal distribution with mean $\mathbf{0}$, and for each predictor $j = 1, \ldots, 5$, the covariance between $\mathbf{x}_{t,j}$ and $\mathbf{x}_{t',j}$ is set to $0.5^{|t-t'|}$ for $t, t' = 1, \ldots, T$, and the between predictor covariance is set to zero. As in the comparisons above, the sample size and the noise level are varied with $n \in \{100, 200, 300\}$ and $r^2 \in \{0.5, 0.7, 0.9\}$.

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. The model complexity parameters for ridge, lasso (a regularization parameter), CTR and LCTR (the iteration number k) were all chosen using 10-fold CV. In order to select the best k, for each training data, we computed the CV SSE for the model size ranging from 1 to k_{max} as described in Sect. 2.6, and then the one with minimum CV SSE is chosen as the best tree size. After the best tree size is obtained via CV, the model is fitted to the entire training data using the best k to obtain the estimated group structure and the associated group coefficients. Then, this final fitted model is applied to the test data to compute the test r^2. Similarly, for each training data set, the regularization parameters for ridge and lasso are obtained using CV, and then this selected regularization parameter value is used when fitting the final model to the training data. For benchmarking with regression trees, on each training data set, the best tree size is also chosen using 10-fold CV. We first divide the data set into 10 equally-sized folds. For the ith hold-out fold, we fit a regression tree using the remaining 9 folds with the number of terminal nodes varying from 1 to 10. Then, a linear regression model is fitted with the number of derived predictors ranging from 1 to 10. For each model, we obtain the CV SSE to predict the ith fold. We repeat this for each hold-out fold ($i = 1, \ldots, 10$) and then sum the CV hold-out SSEs to give a CV SSE for each of the 10 models. Among 10 possible models, we retain only the one that has the smallest CV SSE. We summarize the accuracy of the models over the 100 replicates by averaging the test r^2 values (denoted $\bar{r}^2$).

3.2 Results

Experiment results comparing the accuracy with independent predictors are plotted in Figs. 11, 12, 13, 14. In the figures, columns correspond to different true r^2 values, and different values of n are shown within each panel. The main takeaways from the numerical experiments are as follows. When the coefficients have a true group structure as illustrated in Fig. 8a–b, LCTR outperforms other methods in all cases as shown in Figs. 11, 12 including the high-noise experiments. In all of these experiments, LCTR almost achieves the true r^2 values by correctly identifying the group structure. In addition to being more accurate, LCTR encourages simple and parsimonious models. For example, in Fig. 8a experiment, we compute the mean number of predictors with nonzero-coefficient estimates across 100 replicates. On average, when $n = 1500$ and the true $r^2 = 0.5$, lasso produces 467 predictors with nonzero-coefficient estimates, and when $n = 3000$ and the true $r^2 = 0.9$, lasso produces 959 predictors with nonzero-coefficients (the true coefficients have no sparsity in this example). Ridge

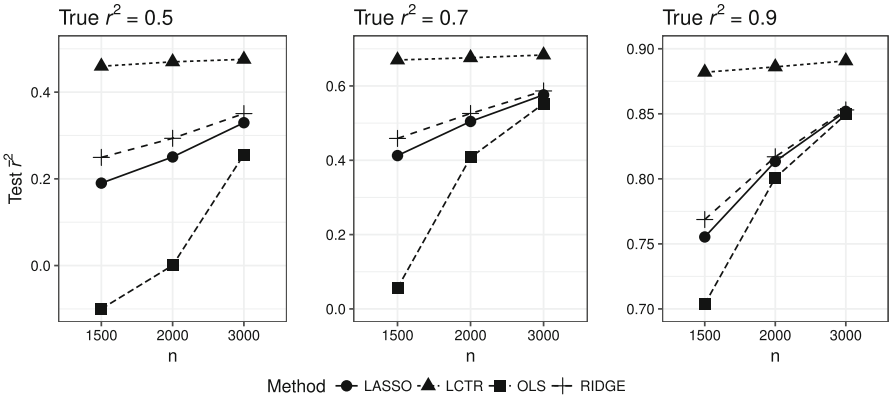


Fig. 11 Simulation results for the experiments illustrated in Fig. 8a

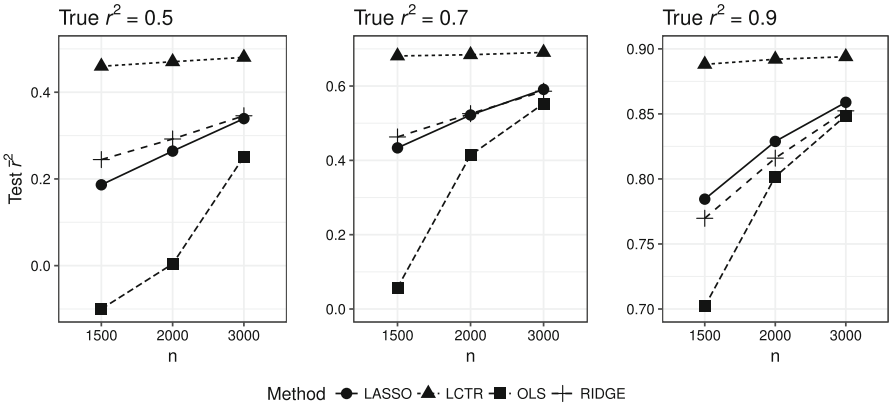


Fig. 12 Simulation results for the experiments illustrated in Fig. 8b

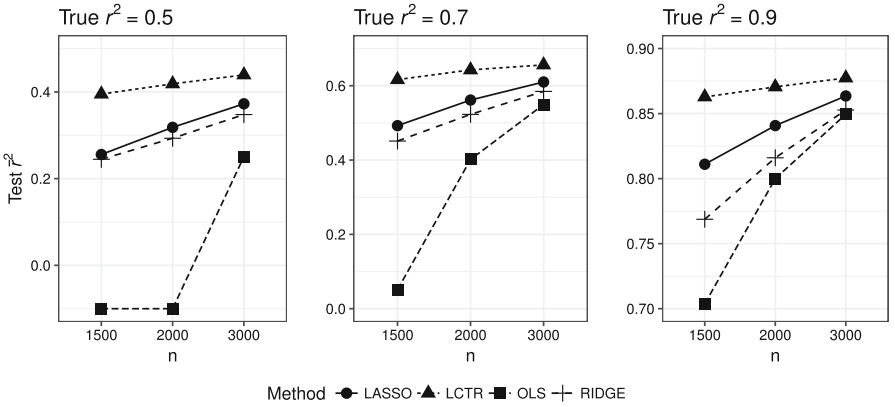


Fig. 13 Simulation results for the experiments illustrated in Fig. 8c

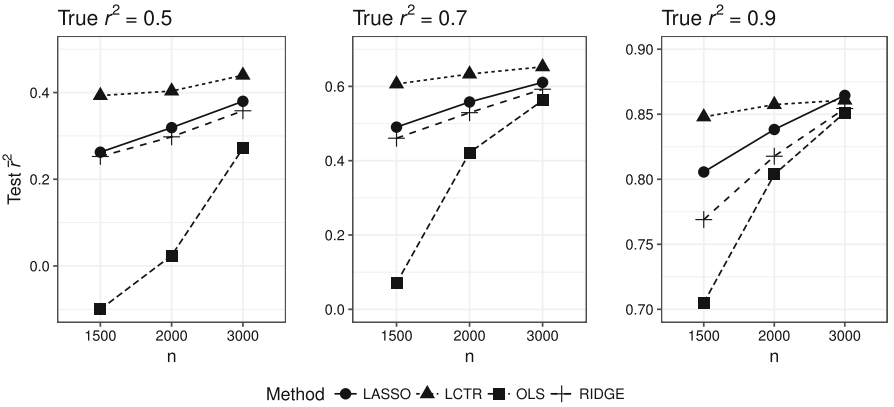
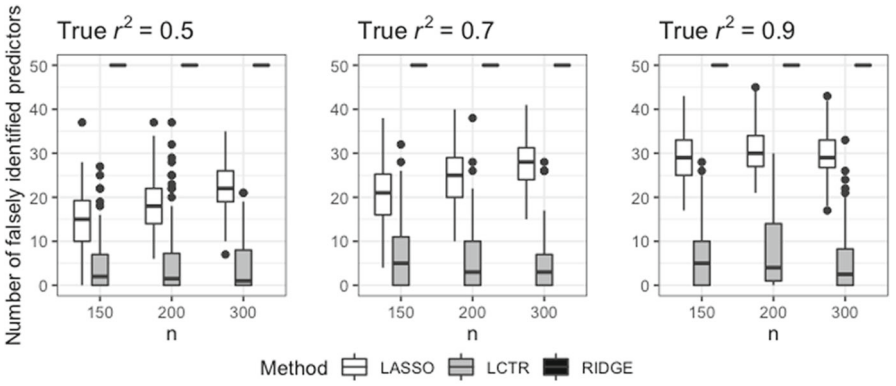


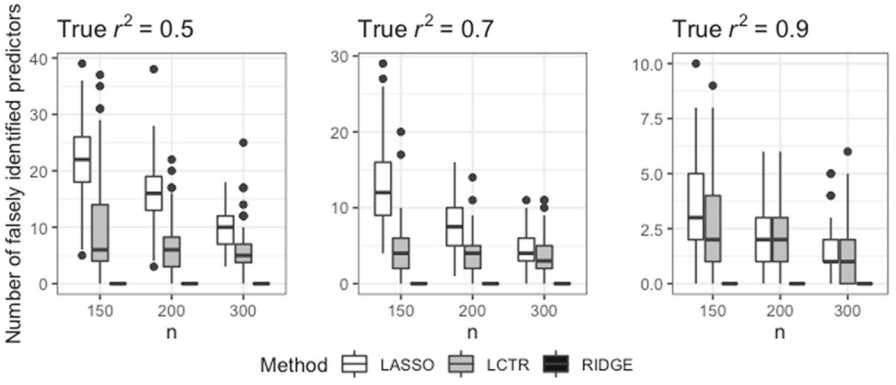
Fig. 14 Simulation results for the experiments illustrated in Fig. 8d

includes all predictors at all replicates. The means of all the estimated coefficients across 100 replicates are nonzero with lasso because of no sparsity in the coefficients in this example. On the other hand, when $n = 1500$ and the true $r^2 = 0.5$, LCTR terminated in 6 iterations on average (obtained via 10-fold CV), and when $n = 3000$ and the true $r^2 = 0.9$, the average number of iterations is 27. We design the Fig. 8c experiment to reflect a real-life situation when the coefficients change smoothly over time based on some exponential and linear curves. Even though the coefficients do not have an explicit group structure, LCTR is superior to other methods in all cases. Finally, similar to the time series data, in Fig. 8d experiment, we enforce a repeating pattern on the coefficients. Even though the split-point search procedure gives priority to the predictors observed at the most recent time points to be included in the model through splitting groups starting from the most recent time point, LCTR achieves the best accuracy levels in all cases as in Fig. 14 except the case when $n = 3000$ and the true $r^2 = 0.9$. In this case, the accuracy is comparable with lasso, which is expected, because half of the coefficients are 0. We terminated the runs when $k_{max} = 60$, and for almost all replicates when $n = 3000$ and the true $r^2 = 0.9$, the CV finds the best k as 60, which indicates that with an increasing k_{max}, LCTR may achieve a better predictive accuracy than lasso for this case as well.

We evaluate the performance of LCTR, lasso, and, ridge to understand whether zero- and nonzero-coefficient predictors are accurately identified. The rows of Fig. 15 show the number of falsely identified zero- and nonzero-coefficient predictors, respectively. Since ridge does not perform any variable selection, all zero-coefficient predictors are falsely identified as nonzero-coefficient predictors and all nonzero-coefficient predictors are correctly identified as expected. On the other hand, LCTR is able to identify more than 80% of the predictor observations correctly in almost all cases, which is far better than lasso. Additionally, we analyze the bias in the coefficients with the medium-size noise level (e.g., $r^2 = 0.7$) and relatively larger sample size $n = 300$.



(a) Number of falsely identified zero-coefficient predictors



(b) Number of falsely identified nonzero-coefficient predictors

Fig. 15 Variable selection performance for the experiment with $p = 5$ and $T = 20$ across 100 replicates

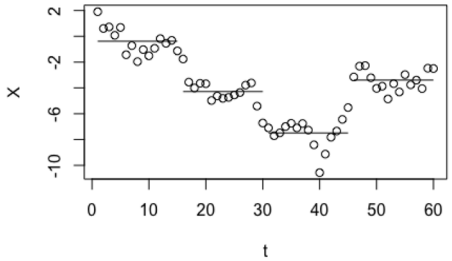
In Fig. 9, lines correspond to the means of the estimated coefficients across 100 replicates. While the lasso penalty shrinks all of the estimated coefficients toward zero, LCTR group structure introduces almost no bias on the coefficient estimates.

Table 1 reports the average computation times (in seconds) across ten replicates of each experiment (there was low replicate-to-replicate variability). The computation times for LCTR, lasso, ridge include 10-fold CV to select their model complexity parameter. In terms of computational expense, LCTR, lasso and ridge are all roughly linear in n and p as shown in Table 1, and LCTR is faster than `glmnet` implementation of lasso and ridge, which is considered extremely fast in the literature.

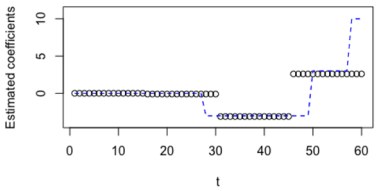
PAA is used to reduce the dimension of longitudinal data and extract predictors in an unsupervised way. Figure 16a illustrates an observation from the data set (i.e., one row of $[\mathbf{x}_{1,1}, \ldots, \mathbf{x}_{T,1}]$) and its four splits produced by PAA. PAA groups predictor observations into four equal-length subsequences, and constructs new predictors in the low-dimensional space by taking the mean of the observations in each subsequence. For a predictive comparison, on each training data, a linear regression model is then

Table 1 Computational time (sec.) comparison of LCTR, lasso, and ridge for various pT and n

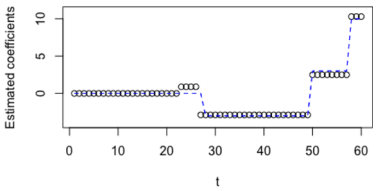
pT	LCTR	Lasso	Ridge
(a) $n = 10^4$ and $p \in \{10, 20, 40, 80\}$ and $T = 50$			
500	5	21	30
1000	11	38	58
2000	26	83	121
4000	59	194	248
n	LCTR	Lasso	Ridge
(b) $pT = 500$ and $n \in \{10^3, 10^4, 10^5, 10^6\}$			
10^3	1	3	4
10^4	5	21	30
10^5	74	208	301
10^6	924	3377	4398



(a) PAA results for a longitudinal predictor. Markers represent the observations of a random walk over $T = 60$, and the horizontal lines correspond to the derived predictors defined as the means of each equal-length subsequence



(b) PAA estimated group structure



(c) LCTR estimated group structure

Fig. 16 Estimated group structure via PAA vs. LCTR for $p = 1$, $T = 60$, and $\beta_{i,1} = 0$ for $i = 1, \ldots, 27$, $\beta_{i,1} = -3$ for $i = 28, \ldots, 49$, $\beta_{i,1} = 3$ for $i = 50, \ldots, 57$, and $\beta_{i,1} = 10$ for $i = 58, \ldots, 60$. Blue dashed lines represent the true coefficients, and the markers represent the estimated coefficients in (b) and (c)

fitted by using the four new predictors produced by PAA, and the estimated coefficients are transformed back to the original 60 dimensional space for illustration. For comparison, LCTR is fitted with $k = 4$ since the number of groups in the true coefficient structure is four. Figure 16b–c demonstrate the coefficient estimates obtained with PAA and LCTR, respectively, for a replicate with the true $r^2 = 0.9$ and $n = 120$. For the PAA results in Fig. 16b, while the most influential predictor observations (i.e., the predictors $[\mathbf{x}_{58,1}, \mathbf{x}_{59,1}, \mathbf{x}_{60,1}]$) are grouped with the ones that have less influence on the response variable, some of the predictors that have a true negative influence on the response variable are placed into the group that has a positive coefficient estimate. In contrast, in LCTR, the length of subsequences varies depending on their effect on the response variable. In this sense, relevant subsequences of predictors are automatically detected, and the ones that have no effect on the response variable are assigned a zero coefficient. Table 2 illustrates the predictive advantage of LCTR over PAA via average test r^2 and MSE values. Since the test r^2 value is a scaled version of (one minus) the test MSE, we provide both the test r^2 and MSE values for only this example as an illustration.

For comparison of regression trees and LCTR, Table 3 summarizes the average test accuracy and the best model size for different true r^2 values and different values of n. While the best model size is always around four with LCTR, for traditional trees,

Table 2 Comparison of PAA and LCTR by averaging the test r^2 values across 100 replicates. Numbers in the parenthesis are the average test MSE values

	r^2			
	n	0.5	0.7	0.9
PAA	60	0.39 (44,332)	0.60 (20,966)	0.80 (8,354)
	90	0.42 (42,395)	0.62 (20,243)	0.80 (8,095)
	120	0.42 (41,935)	0.62 (20,007)	0.81 (7,930)
LCTR	60	0.40 (43,389)	0.65 (18,493)	0.87 (5,166)
	90	0.44 (40,525)	0.67 (17,503)	0.88 (4,856)
	120	0.46 (39,510)	0.68 (16,994)	0.88 (4,781)

Table 3 Comparison of regression tree and LCTR by averaging the test r^2 values across 100 replicates. Numbers in the parenthesis are the average number of derived predictors at the best model

r^2	Regression tree			LCTR		
n	0.5	0.7	0.9	0.5	0.7	0.9
60	0.29 (8.4)	0.48 (8.7)	0.72 (9.1)	0.37 (2.9)	0.59 (4.3)	0.87 (5.0)
90	0.35 (8.6)	0.55 (9.0)	0.76 (9.1)	0.41 (3.9)	0.64 (4.3)	0.88 (5.1)
120	0.38 (8.6)	0.57 (9.1)	0.76 (9.2)	0.43 (4.2)	0.66 (4.4)	0.89 (5.4)

Table 4 Comparison of CTR and LCTR by averaging the test r^2 values across 100 replicates for two different true $\boldsymbol{\beta}$ structures

	r^2	CTR			LCTR		
	n	0.5	0.7	0.9	0.5	0.7	0.9
Experiment 1	100	0.23	0.48	0.75	0.32	0.56	0.84
	200	0.35	0.58	0.84	0.40	0.63	0.87
	300	0.39	0.61	0.86	0.43	0.65	0.88
Experiment 2	100	0.26	0.48	0.76	0.33	0.57	0.84
	200	0.36	0.58	0.83	0.40	0.63	0.87
	300	0.39	0.61	0.86	0.43	0.65	0.88

the best test accuracy is obtained with a larger number of derived predictors. Moreover, the overall predictive accuracy of LCTR is consistently better than predictive accuracy for traditional trees. Since the true group structure is not correctly identified with traditional trees, and they split the predictors into smaller pieces, this reduces interpretability in addition to worsening predictive accuracy.

Table 4 summarizes the predictive accuracy of CTR and LCTR models by averaging the test r^2 values across 100 replicates. The best iteration number k is chosen via CV for both CTR and LCTR using $k_{max} = 20$. For both of the true models, LCTR performs significantly better than CTR. To illustrate the interpretability advantages of LCTR over CTR, Fig. 10 shows the estimated coefficients for a replicate with $n = 300$ and $r^2 = 0.9$, which is the setting in which both methods perform best. CTR is not able to capture the temporal characteristic of the true coefficients, and coefficient estimates (illustrated with markers) do not follow the true group structure. On the other hand, with LCTR, the coefficient estimates for each predictor are contiguous in time. Therefore, even though the predictor observations at the closer time points have a similar effect on the response variable, the CTR's split-point search procedure includes those predictor observations into different groups with different coefficients, and thus the predictors in the same group are not necessarily contiguous in time. On other other hand, LCTR's split-point search procedure considers potential splits on both time and on the predictors to preserve the interpretability and the predictive accuracy.

We finally evaluate LCTR for examples in which there are both longitudinal and non-longitudinal predictors. For this, we use the residential building data set from the UCI Machine Learning Repository (Rafiei and Adeli 2016). The data set consists of 8 physical and financial non-longitudinal predictors and 19 economic predictors, each of which is measured at five different time points (thus comprising $5 \times 19 = 95$ predictor observations) resulting in a total of 103 predictors. We consider two different regression settings, in which the actual sales price and actual construction cost are the two response variables. The data set consists of 372 observations, and we randomly 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

Table 5 Predictive performance comparison via average r^2 value for the residential building data set with two different response variables. The numbers in the parentheses are the average number of groups in the final model across 100 replicates

n	LCTR	Lasso	Ridge
(a) Actual sales prices			
74	0.96 (6)	0.96	0.49
186	0.97 (8)	0.97	0.95
298	0.97 (9)	0.98	0.96
(b) Actual construction cost			
74	0.94 (6)	0.95	0.72
186	0.96 (7)	0.96	0.94
298	0.96 (6)	0.96	0.94

100 different random splits of the data into training and test sets of respective sizes and then computed the average test r^2, averaged across the 100 replicates. Table 5 summarizes the predictive accuracy of LCTR, lasso, and ridge. Although LCTR and lasso perform comparably well, ridge does not perform well when the sample size is small. This indicates that a small number of predictors may be enough to explain both of the response variables, and since ridge is unable to do variable selection, this may explain its poor performance for small sample size. On the other hand, LCTR is able to discover important predictors within a relatively small number of groups indicated by the numbers in parentheses in Table 5. Regarding interpretability of the LCTR model, we examined the group structure of an LCTR model fit to all 372 training observations. To predict the actual construction cost, the preliminary estimated construction cost and the duration of construction are the two non-longitudinal predictors included into the final model. In addition to those two predictors, the project locality and the price of a unit at the beginning of the project are two other non-longitudinal predictors that are influential to predict the actual sales price. The remaining groups include the longitudinal predictors.

Additional experiments using two publicly available data sets with longitudinal predictors are provided in Appendix 3 to compare the predictive performance of LCTR, lasso, and ridge.

4 Discovering temporal pattern: engagement with news media

We now illustrate LCTR on a real data set studying local news engagement, highlighting the interpretability of the model. Media websites that offer news are shifting away from relying on advertising revenues toward user-supported subscription models. In the past, such sites would attract a large audience with low subscription fees and then sell access to the audience to advertisers. Today media platforms, primarily Google and Facebook, mediate news content and direct consumers to newspaper stories. These platforms take an increasingly large share—up to 70% (Sterling 2019)—of advertising revenue. The result is that the organizations that create news content face declining advertising revenue, thus limiting their capacity to produce news. Newsroom

employment has plummeted by 46.7% between 2008 and 2018 (Pew Research Center for Journalism and Media 2019). One consequence is "news deserts," in which 1,300 communities have been left with no news coverage, and 2000 of the 3,143 counties in the US have no daily newspaper (Abernathy 2018). A promising strategy is to increase subscription fees so that readers are underwriting a much large share of the cost to produce news. This strategy, however, depends on creating unique, differentiated content for which consumers are willing to pay (WTP).

At the same time, news organizations now have access to better data on reading behaviors than ever before. Click stream data records every page that every user views over time, as well as contextual information such as the device and browser used, location of the session, and the amount of advertising. We use the *regularity* of reading as a proxy for WTP, where regularity is measured by the number of days in a month that a subscriber reads at least some content. Regularity is a reflection of a reading habit and is a good leading indicator of customer retention and WTP (Kim et al. 2021). Such models could provide editors with valuable insights for enhancing reader engagement with the content, which, in turn, increases regularity and WTP. Building such predictive models, however, is not straightforward because the number of possible pieces of content a user can view is very large and page-view (PV) counts per article are sparse (small counts). While there is a large literature on using PV data to measure user engagement (Lalmas et al. 2014), extant methods have not made the link to regularity and WTP. One conclusion that can be drawn from the literature is that not all PVs are equal. Some are highly engaging, others are less engaging, and some may even disengage the reader (Lu et al. 2019; Miroglio et al. 2018). Identifying which reading behaviors are highly engaging and disengaging is important to understand the drivers of regularity and retain subscribers.

We analyze data from a local news site located in one major US city. Our universe for the analysis is all households that had digital-only subscriptions to the paper as of September 15, 2019. The response variable is the number of days that a subscriber read any content during the last four-week period (September 16–October 14), and thus takes values 0–28. The predictors consist of the PV counts of 47 topics or contextual experiences during the previous 48 weeks, giving a total of $48 \times 47 = 2256$ predictors. We used $k_{max} = 20$ and 10-fold CV, and obtained the best iteration number $k = 8$. The tree structure is given in Fig. 17. The final group structure is given below:

- Derived predictor 1 ($\hat{\alpha}_{8,1} = 0.20$) with $G_{8,1} = \{\{$college basketball, college football, obituaries, homepage views, location 1, location 2, location 3, location 4, source Google search, source direct, source email, source social media, source Bing/Yahoo/AOL, source Legacy.com, source Google search, source newsletter$\}$, $[45, 47]\}$
- Derived predictor 2 ($\hat{\alpha}_{8,2} = 0.68$): $G_{8,2} = \{\{$college basketball, obituaries, homepage views, location 1, location 2, location 3, location 4, source Google search, source direct, source email, source social media, source Bing/Yahoo/AOL, source Legacy.com, source Google search$\}$, $[48, 48]\}$
- Derived predictor 3 ($\hat{\alpha}_{8,3} = 0.12$) with $G_{8,3} = \{\{$college football, source newsletter$\}$, $[48, 48]\}$

- Derived predictor 4 ($\hat{\alpha}_{8,4} = -0.20$) with $G_{8,4} = \{\{$crime, other sports, US news, state, opinion, college news, living, life/culture, real estate, outdoor, pro football, health, baseball, Ncomment, posts to social media, source Google News$\}$, [45, 48]$\}$
- Derived predictor 5 ($\hat{\alpha}_{8,5} = 0.19$) with $G_{8,5} = \{\{$news, business, politics, high school sports, entertain, weather, schools, restaurant/food, celebrity, state fair, traffic, ad blocker$\}$, [45, 48]$\}$
- Derived predictor 6 ($\hat{\alpha}_{8,6} = 0.26$) with $G_{8,6} = \{\{$crime, news, obituaries, schools, source email, source Bing/Yahoo/AOL, source newsletter$\}$, [40, 44]$\}$
- Derived predictor 7 ($\hat{\alpha}_{8,7} = -1.53$) with $G_{8,7} = \{\{$politics, US news, restaurant/food, real estate, pro football$\}$, [44, 44]$\}$
- Derived predictor 8 ($\hat{\alpha}_{8,8} = 0.39$) with $G_{8,8} = \{\{$college basketball, other sports, US news, weather, college news, restaurant/food, life/culture, celebrity, state fair, health, traffic, posts to social media, location 4, source direct, source social media, source Google News$\}$, [42, 43]$\}$
- Derived predictor 9 ($\hat{\alpha}_{8,9} = -0.33$) with $G_{8,9} = \{\{$college football, business, politics, US news, high school sports, entertain, real estate, state fair, traffic, baseball, posts to social media, homepage views$\}$, [40, 41]$\}$

Derived predictor 2 has the largest positive coefficient (0.68) and mainly consists of local topics during the most recent week (48). The city has a university with popular basketball and football teams. The news organization provides some of the best coverage of these teams, and reading articles about them is associated with more reading in the future. Other engaging topics include obituaries and home PVs, where subscribers are likely scanning headlines for an update. PVs from four locations are also included, indicating that readers who are not from out of state (location 5) tend to read more next month. Certain sources are better than others. Derived predictor 1 has the same variables, but from an earlier time period (weeks 45–47) and with a smaller coefficient (0.20). LCTR has thus determined that it should give more weight to recent behavior (week 48) than earlier behavior. We do not see variables from weeks 1–39, suggesting that roughly two months of data is sufficient to predict future reading regularity. Derived predictor 3 assigns even more weight (0.12) to stories about the local college football team, and readers who come from clicks on newsletters, which are an important way for news organizations to drive traffic to their sites without being dependent on the page ranking algorithms used by Google, Facebook and other social/search platforms. Derived predictors 5 and 6 identify other topic areas that drive engagement for the news organization, although with smaller coefficient (0.19 and 0.26, respectively).

Not all content engages readers, and some drives them away. The editor should be alerted to these areas so that alternative approaches of covering them could be tried. Derived predictor 4 has a negative association (-0.20) with regularity next month. Topics such as US news, professional football and baseball, and health are covered in many other news outlets, probably with greater resources and higher quality. For example, US news is covered by cable and network TV, news magazines, national newspapers, etc. The local news organization has no competitive advantage in covering such topics, other than perhaps discussing local implications of national news. It might be better for the local news organization not to try to cover such topics unless there is a local angle. Similar statements may be true for living, life/culture, and outdoor. Crime

is most likely local and not covered elsewhere, but the negative slope suggests that the way it is covered might be off-putting. State and news about the local college should be strengths since they are local topics, but reading them seems to drive users away. Derived predictor 7 makes the negative association between some of these variables and future regularity even more negative (−1.53), using data from an earlier week (44).

Derived predictors 8 ($\hat{\alpha} = +0.39$) and 9 ($\hat{\alpha} = -0.33$) include predictor observations from earlier weeks. Some of those on derived predictor 9 have opposite signs, such as college football, which has a positive coefficient during the weeks 45–47, and a negative coefficient during the weeks 40–41. While including a core, local topic like college football on a predictor with a negative coefficient could be spurious, the editor might also consider that weeks 40–41 are July 15–28, which is prior to the start of the football season (officially August 24, 2019) when there is less coverage of the topic. College football has a positive coefficient during weeks 45–47 (August 19–September 8). Preseason football coverage should be examined.

To analyze the predictive performance of LCTR, we split the data into training and test sets, and the size of the test set is fixed at 336 (i.e, 20% of the data set with the sample size of 1676). We repeat this process for 30 random splittings of the data into training and test sets, and compute the average test r^2. The average test r^2 values are 0.49, 0.51, and 0.51 for LCTR, lasso, and ridge, respectively. In this example, lasso and ridge perform slightly better than LCTR in terms of predictive accuracy. However, LCTR is parsimonious with small number of groups and provides insight into the relationship between predictors via its tree structure. While LCTR has nine derived predictors in the final model, lasso and ridge include 162 and 2042 predictors, respectively, in their final model. The computation times including 10-fold CV for LCTR, lasso, and ridge are 2, 5, and 8 s, respectively.

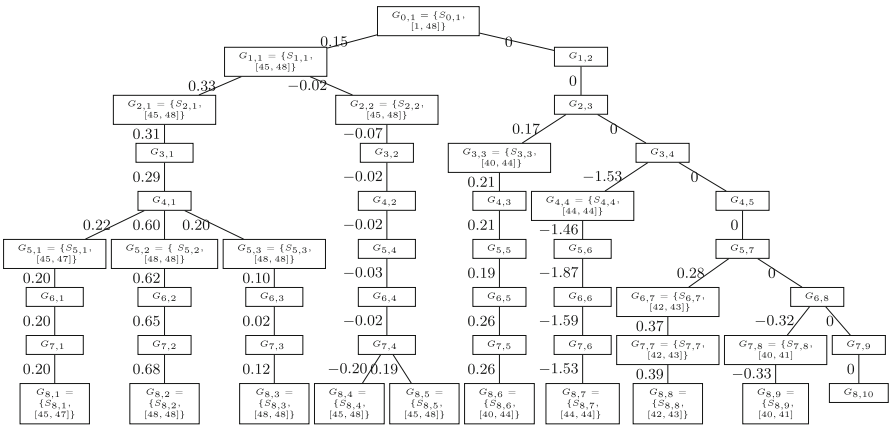


Fig. 17 Illustration of the LCTR-estimated hierarchical group structure and coefficients in the news engagement example

5 Conclusion

In this paper, we develop a novel CTR algorithm for longitudinal data (LCTR) to identify temporal patterns in high-dimensional multivariate longitudinal data sets. LCTR leads to a simple, highly-interpretable tree structure via successively partitioning the subsequences of predictor observations that have the similar effect on the response variable into the groups. Finding such groups of predictors that share a common regression coefficient is an automated way of feature engineering and selection where the sum of predictors within each group represents a new derived predictor. The simulation results demonstrated that the prediction performance is superior to ridge, lasso, and OLS because the LCTR algorithm considers the time component during the search procedure. In addition to being more accurate, LCTR can be considered as a faster alternative to the regularized regression methods such as ridge and lasso.

There are many opportunities for future research. One line of future research could be to develop ways to include nonlinear transformations of the observed and/or derived variables. One way of doing this is to allow nonlinear functions of derived predictors, where the derived predictors are still sums over contiguous time intervals. A related extension would be to search for interactions between derived variables. We have focused on the situation where each row of the regression model represents a different sampling unit. A similar approach can be developed for the situation with temporal **y** values, where **y** is a time series with each row of the regression model representing the response at different times, for either a common sampling unit or for different sampling units. In this case, since past response values serve as additional predictors, they should be included along with other predictors in the split-point search. Moreover, modifications of CV appropriate for regressions with correlated ε would have to be used to select k. While being promising for temporal data sets, the LCTR method can be further extended to other longitudinal studies with spatial patterns. In this case, the split-point search procedure should be modified to search across multiple domains. With more sensors and digital environments, all of these types of data will become more common in the future, increasing the need for automated supervised learning procedures like LCTR.

Declarations

Conflict of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

Appendix 1: Proofs

Theorem 1 During iteration k, suppose we consider splitting group $G_{k-1,l} = \{S_{k-1,l}, [t^b_{k-1,l}, t^e_{k-1,l}]\}$ at temporal cutpoint t, and search for the optimal horizontal split point j_t that most reduces the SSE when augmenting the basis from $\mathbf{Z}_{k-1}$ to $[\mathbf{Z}_{k-1}, \mathbf{s}_{t,j_t}]$ such that $\mathbf{s}_{t,j_t} = \sum_{t'=t}^{t^e_{k-1,l}} \mathbf{x}_{t',j_t}$. The derived predictor $\mathbf{s}_{t,j_t}$ is obtained as the maximizer

of the following mathematical model

$$\mathbf{s}_{t,j_t} = \arg\max_{\{\mathbf{s}_{t,j}:\sum_{t'=t}^{t^e_{k-1,l}} \mathbf{x}_{t',j},\, \forall j \in S_{k-1,l}\}} R(\mathbf{s}_{t,j}), \tag{12}$$

and among the set $\{\mathbf{s}_{t,j} : \sum_{t'=t}^{t^e_{k-1,l}} \mathbf{x}_{t',j},\ \forall j \in S_{k-1,l}\}$ of sums, $\mathbf{s}_{t,j_t}$ is the one with the largest squared partial coefficient ρ_{t,j_t}.

Proof We know that the reduction $R(\mathbf{z}_{new})$ in the SSE when the derived predictors grow from $\mathbf{Z}_{k-1}$ to $\left[\mathbf{Z}_{k-1},\, \mathbf{z}_{new}\right]$ is computed via (10), where $\mathbf{z}_{new} = \mathbf{z}_{old} + \mathbf{s}_{t,j} = \mathbf{z}_{old} + \sum_{t'=t}^{t^e_{k-1,l}} \mathbf{x}_{t',j}$. For the case when augmenting the basis from $\mathbf{Z}_{k-1}$ to $[\mathbf{Z}_{k-1},\, \mathbf{s}_{t,j}]$, let $\mathbf{z}_{new} = \mathbf{s}_{t,j}$ and $\mathbf{z}_{old} = \mathbf{0}$. In such a case, $N_{old} = 0$, $D_{old} = 0$, and $\mathbf{e}_{\mathbf{z}_{old}} = \mathbf{0}$, and plugging these values into (10), we obtain

$$R(\mathbf{s}_{t,j}) = \frac{\left(\mathbf{e}_{\mathbf{y}}^{\mathsf{T}} \sum_{t'=t}^{t^e_{k-1,l}} \mathbf{e}_{t',j,k-1}\right)^2}{\left(\sum_{t'=t}^{t^e_{k-1,l}} \mathbf{e}_{t',j,k-1}\right)^{\mathsf{T}} \left(\sum_{t'=t}^{t^e_{k-1,l}} \mathbf{e}_{t',j,k-1}\right)}. \tag{13}$$

From (13), we find that $R(\mathbf{s}_{t,j}) = \rho_{t,j}$. Therefore, among the set $\{\mathbf{s}_{t,j} : \sum_{t'=t}^{t^e_{k-1,l}} \mathbf{x}_{t',j},\ \forall j \in S_{k-1,l}\}$ of sums, the sum $\mathbf{s}_{t,j_t}$ with the largest squared partial correlation ρ_{t,j_t} maximizes the reduction $R(\mathbf{s}_{t,j_t})$ in the SSE. In other words, $\rho_{t,j_t} \geq \rho_{t,j}$ for $j \in S_{k-1,l}$. □

Appendix 2: Efficient model updating

For computational purposes, after finding the new derived predictor $\mathbf{z}$ ($\mathbf{z} = \mathbf{z}_{k,2k-1}$ when splitting a zero-coefficient group or $\mathbf{z} = \mathbf{z}_{k,\tilde{k}}$ when splitting a nonzero-coefficient group as explained in Sect. 2.4) at the end of the kth iteration, we update the model as follows. Let the (t,j)th column of the matrix $\mathbf{E}_k$ (the column that corresponds to jth predictor at time t) be the cumulative sum of errors from the most recent time point T to the time point t, i.e., $\mathbf{E}_k$ is defined such that its (t,j)th column is

$$\begin{aligned}(\mathbf{E}_k)_{t,j} &= \sum_{t'=t}^{T} \mathbf{e}_{t',j,k} = \sum_{t'=t}^{T} \mathbf{x}_{t',j} - \mathbf{P}_k \sum_{t'=t}^{T} \mathbf{x}_{t',j} \\ &= \sum_{t'=t}^{T} \mathbf{e}_{t',j,k-1} - \frac{\mathbf{e}_{\mathbf{z},k-1}^{\mathsf{T}} \sum_{t'=t}^{T} \mathbf{e}_{t',j,k-1}}{\mathbf{e}_{\mathbf{z},k-1}^{\mathsf{T}} \mathbf{e}_{\mathbf{z},k-1}} \mathbf{e}_{\mathbf{z},k-1},\end{aligned} \tag{14}$$

and $(\mathbf{w}_k)_{t,j} = \sum_{t'=t}^{T} \mathbf{e}_{t',j,k}^{\mathsf{T}} \mathbf{y}$ and $(\mathbf{u}_k)_{t,j} = \left(\sum_{t'=t}^{T} \mathbf{e}_{t',j,k}\right)^{\mathsf{T}} \sum_{t'=t}^{T} \mathbf{e}_{t',j,k}$. We update and store the $n \times pT$ matrix $\mathbf{E}_k$ and the $pT \times 1$ vectors $\mathbf{w}_k$ and $\mathbf{u}_k$ at the end of iteration k

using the relationship between the cumulative errors at iteration $k-1$ versus at iteration k in an efficient manner. The $n \times pT$ matrix $\mathbf{E}_k$ can be updated from $\mathbf{E}_{k-1}$ via

$$\mathbf{E}_k = \mathbf{E}_{k-1} - \frac{\mathbf{e}_{\mathbf{z},k-1}}{\mathbf{e}_{\mathbf{z},k-1}^{\mathsf{T}}\mathbf{e}_{\mathbf{z},k-1}}\mathbf{v}_k^{\mathsf{T}}, \tag{15}$$

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

$$\mathbf{w}_k = \mathbf{w}_{k-1} - \mathbf{v}_k \frac{\mathbf{e}_{\mathbf{z},k-1}^{\mathsf{T}}\mathbf{y}}{\mathbf{e}_{\mathbf{z},k-1}^{\mathsf{T}}\mathbf{e}_{\mathbf{z},k-1}} \text{ and } \mathbf{u}_k = \mathbf{u}_{k-1} - \frac{\mathbf{v}_k \circ \mathbf{v}_k}{\mathbf{e}_{\mathbf{z},k-1}^{\mathsf{T}}\mathbf{e}_{\mathbf{z},k-1}}, \tag{16}$$

which follows from their definitions. Here, $\circ$ denotes the element-wise product. The updates (15)–(16) provide an efficient means of updating $\mathbf{E}_k$, $\mathbf{w}_k$ and $\mathbf{u}_k$ at the end of iteration k to be used during the split-point search at iteration $k+1$ to compute the reduction $R(\mathbf{z}_{new})$ in (10). In the case of splitting an existing group, we apply the updates (15)–(16) twice because the number of derived predictors effectively increases by two as explained in Sect. 2.4 (see lines 12–16 in Algorithm 1 for updating the model).

Appendix 3: Additional experiments

In this section, we compare the predictive performance of LCTR with ridge and lasso using publicly available real data sets. We first use the weekly sales transaction data set (Tan 2017) from the UCI Machine Learning Repository. The data set contains weekly purchased quantities of 811 products over 52 weeks. For each product, the response is the maximum number of sales during the last four-week period, and predictors consist of sales during the previous 48 weeks. About half of the products are selected randomly to serve as the training set (e.g., 405 products) and the remaining half serves as the test set (e.g., 406 products). We repeated this process for 30 different random splittings of the data into training and test sets, and computed the average test r^2 value across all 30 replicates. All methods performed comparably well in terms of predictive accuracy with the resulting average test r^2 values being approximately 0.94 for LCTR, lasso, and ridge. Figure 18 presents the estimated coefficients obtained with LCTR, lasso, and ridge in their final model fitted with all 811 products. For this real data example, we do not know the ground truth regarding which predictors are most influential. However, from Fig. 18, both lasso and ridge find that predictors between time points 20 and 30 are the most influential ones. As pointed out in Sect. 2, LCTR is able to provide a close piecewise approximation to both lasso and ridge estimates using in this case only five pieces (i.e., five groups) without losing predictive advantage even though the distant past influences the response more strongly than the recent past.

Next, we consider the electricity load diagrams data set (Trindade 2015) also from the UCI Machine Learning Repository. The data set contains the electricity consumption of $n = 370$ clients recorded every 15 min. In this experiment, we compare the

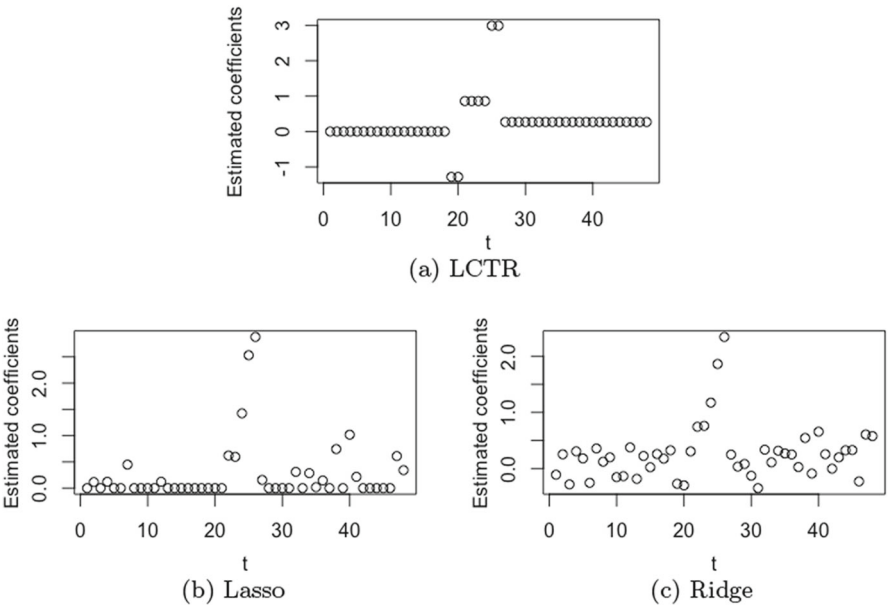


Fig. 18 Estimated coefficients for the weekly sales transaction data set

predictive performance of LCTR, lasso, and ridge using the 2014 electricity consumption data such that the response is constructed from the data observed in December, and predictors are constructed from the data observed from January to November. To check the performance for varying numbers of predictors (i.e., for $n > p$, $n \approx p$, and $n < p$ cases), we considered three different regression settings, in which the predictors are generated at three different granularity levels. First, the predictors consist of the weekly electricity consumption of a client during the initial 48 weeks. Next, we consider the daily electricity consumption from January to November with a total of 334 predictors per client. Finally, the energy consumed per 12 h is taken as a predictor to construct a total of 668 predictors. For these three different regressions, the corresponding response variables are taken to be the maximum weekly, daily, and 12-hour electric consumption during December, respectively. For each of these three regressions, 370 observations are randomly split into training and test sets with an 80:20 ratio repeated for a total of 30 different random splittings, and the average test r^2 values are given in Table 6. In this example, LCTR and lasso consistently perform better than ridge (LCTR performs slightly better than lasso, although both have r^2 close to 1.0), and coefficient estimates in the final models of LCTR and lasso indicate that while only a few predictors observed at the most recent time points have substantial effects, the remaining predictors have almost no effect on the response. As a possible explanation for the slightly better performance of LCTR versus lasso, although lasso is able to select the influential predictors, it also tends to include some predictors that may not be influential due to the noise in the data. Ridge includes all predictors in the model, which may explain why its average test accuracy is always lower. For this

Table 6 Predictive performance comparison via average test r^2 value for the electricity load diagrams data set. The numbers in the parentheses are the average number of groups for LCTR and the average number of nonzero-coefficient predictors for lasso and ridge in the final model across 30 replicates

	$T \times p$	LCTR	Lasso	Ridge
Weekly	48	1.00 (4.5)	0.99 (6.1)	0.90 (48)
Daily	334	0.99 (1.6)	0.97 (14.7)	0.91 (334)
12-hour	668	0.98 (1.9)	0.97 (15.6)	0.87 (668)

situation, LCTR's model building procedure allows parsimonious models with a small number of groups, and its split-point search encourages the most recent points to be in these groups. As a result, it predicts the response with near-perfect accuracy. We also note that strong multicollinearity is present between the predictors of both data sets considered in this section, and LCTR produces groups with easy-to-interpret group structure.

References

Abernathy PM (2018) The expanding news desert. University of North Carolina Press, Chapel Hill, NC

Balakrishnan S, Madigan D (2006) Decision trees for functional variables. In: Sixth international conference on data mining (ICDM'06), pp 798–802

Baydogan MG, Runger G (2015) Learning a symbolic representation for multivariate time series classification. Data Min Knowl Discov 29(2):400–422

Baydogan MG, Runger G (2016) Time series representation and similarity based on local autopatterns. Data Min Knowl Discov 30(2):476–509

Baydogan MG, Runger G, Tuv E (2013) A bag-of-features framework to classify time series. IEEE Trans Pattern Anal Mach Intell 35(11):2796–2802

Belli E, Vantini S (2022) Measure inducing classification and regression trees for functional data. Stat Anal Data Min ASA Data Sci J 15(5):553–569

Bertsimas D, Paskov A (2022) World-class interpretable poker. Mach Learn 111(8):3063–3083

Blanquero R, Carrizosa E, Molero-Río C, Romero Morales D (2023) On optimal regression trees to detect critical intervals for multivariate functional data. Comput Oper Res 152:106152

Bondell HD, Reich BJ (2008) Simultaneous regression shrinkage, variable selection and clustering of predictors with OSCAR. Biometrics 64(1):115–123

Breheny P HJ (2009) Penalized methods for bi-level variable selection. Stat Interface 2(3):369–380

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

Carrizosa E, Mortensen LH, Romero Morales D, Sillero-Denamiel MR (2022) The tree based linear regression model for hierarchical categorical variables. Expert Syst Appl 203:117423

Chakrabarti K, Keogh E, Mehrotra S, Pazzani M (2002) Locally adaptive dimensionality reduction for indexing large time series databases. ACM Trans Database Syst (TODS) 27(2):188–228

Dettling M, Bühlmann P (2004) Finding predictive gene groups from microarray data. J Multivar Anal 90(1):106–131

Dietterich TG (2002) Machine learning for sequential data: a review. In: Structural, syntactic, and statistical pattern recognition, pp 15–30

Eiras-Franco C, Guijarro-Berdiñas B, Alonso-Betanzos A, Bahamonde A (2019) A scalable decision-tree-based method to explain interactions in dyadic data. Decis Support Syst 127:113141

Friedman J, Hastie T, Tibshirani R (2010) Regularization paths for generalized linear models via coordinate descent. J Stat Softw 33(1):1–22

Geurts P (2001) Pattern extraction for time series classification. In: Principles of data mining and knowledge discovery. Springer, Berlin, Heidelberg, pp 115–127

Goodman B, Flaxman S (2017) European Union regulations on algorithmic decision-making and a right to explanation. AI Mag 38(3):50–57
Huang J, Ma S, Xie H, Zhang C-H (2009) A group bridge approach for variable selection. Biometrika 96(2):339–355
Ke ZT, Fan J, Wu Y (2015) Homogeneity pursuit. J Am Stat Assoc 110(509):175–194
Kim SJ, Zhou Y, Malthouse ECa (2021) In search for an audience-supported business model for local newspapers: jindings from clickstream and subscriber data. Digit Journal
Lalmas M, O'Brien H, Yom-Tov E (2014) Measuring user engagement. Synth Lect Inf Concepts Retr Serv 6(4):1–132
Laurinec P (2018) TSrepr R package: time series representations. J Open Source Softw
Lin J, Keogh E, Wei L, Lonardi S (2007) Experiencing sax: a novel symbolic representation of time series. Data Min Knowl Discov 15(2):107–144
Lin J, Keogh E, Lonardi S, Chiu B (2003) A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the 8th ACM SIGMOD workshop on research issues in data mining and knowledge discovery. DMKD '03. Association for Computing Machinery, New York, NY, USA, pp 2–11
Lu H, Zhang M, Ma W, Wang C, xia F, Liu Y, Lin L, Ma S (2019) Effects of user negative experience in mobile news streaming. In: Proceedings of the 42Nd international ACM SIGIR conference on research and development in information retrieval. SIGIR'19. ACM, New York, NY, USA, pp 705–714
Miroglio B, Zeber D, Kaye J, Weiss R (2018) The effect of ad blocking on user engagement with the web. In: Proceedings of the 2018 World Wide Web Conference. WWW '18. World Wide Web Conferences, Geneva, Switzerland, pp 813–821
Möller A, Tutz G, Gertheiss J (2016) Random forests for functional covariates. J Chemom 30(12):715–725
Mueen A, Keogh E, Young N (2011) Logical-shapelets: an expressive primitive for time series classification. In: Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining. KDD '11. Association for Computing Machinery, New York, NY, USA, pp 1154–1162
Pew Research Center for Journalism and Media: Newspapers Fact Sheet (July 9, 2019)
Rafiei MH, Adeli H (2016) A novel machine learning model for estimation of sale prices of real estate units. J Constr Eng Manag 142(2):04015066
Rai A (2020) Explainable AI: from black box to glass box. J Acad Mark Sci 48(1):137–141
Rudin C (2018) Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nat Mach Intell 1:206–215
Sterling G (2019) Almost 70% of digital ad spending going to google, facebook, amazon, says analyst firm. Marketingland.com. Retrieved from https://marketingland.com/almost-70-of-digital-ad-spending-going-to-google-facebook-amazon-says-analyst-firm-262565
Sürer O, Apley DW, Malthouse EC (2021) Coefficient tree regression for generalized linear models. Stat Anal Data Min ASA Data Sci J 14:407–429
Sürer O, Apley DW, Malthouse EC (2021) Coefficient tree regression R package. https://github.com/ozgesurer/CTR.git. Accessed 10 June 2023
Sürer O, Apley DW, Malthouse EC (2021) Coefficient tree regression: fast, accurate and interpretable predictive modeling. Mach Learn 1–38
Sürer O, Apley DW, Malthouse EC longitudinal coefficient tree regression R package. https://github.com/ozgesurer/LongCTR. Accessed 11 sep 2022
Tan J (2017) Sales Transactions Dataset Weekly. UCI Machine Learning Repository. https://doi.org/10.24432/C5XS4Q
Team RC (2017) R: A language and environment for statistical computing. R foundation for statistical computing. R Foundation for Statistical Computing, Vienna, Austria
Therneau T, Atkinson B (2019) Rpart: recursive partitioning and regression trees. R package version 4.1-15. https://CRAN.R-project.org/package=rpart
Tibshirani R, Saunders M, Rosset S, Zhu J, Knight K (2005) Sparsity and smoothness via the fused lasso. J R Stat Soc Ser B (Stat Methodol) 67(1):91–108
Trindade A (2015) Electricity load diagrams 2011–2014. UCI Machine Learning Repository. https://doi.org/10.24432/C58C86
Wang L, Chen G, Li H (2007) Group SCAD regression analysis for microarray time course gene expression data. Bioinformatics 23(12):1486–1494
Wang J-L, Chiou J-M, Müller H-G (2016) Functional data analysis. Annu Rev Stat Appl 3(1):257–295

Ye L, Keogh E (2011) Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. Data Min Knowl Discov 22(1):149–182

Ye L, Keogh E (2009) Time series shapelets: a new primitive for data mining. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining. KDD '09. Association for Computing Machinery, New York, NY, USA, pp 947–956

Yuan M, Lin Y (2006) Model selection and estimation in regression with grouped variables. J R Stat Soc Ser B (Stat Methodol) 68(1):49–67

Zhao P, Rocha G, Yu B (2009) The composite absolute penalties family for grouped and hierarchical variable selection. Ann Stat 37(6A):3468–3497

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

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.