Journal of Computational and Graphical Statistics

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/ucgs20

Identifying Heterogeneous Effect Using Latent Supervised Clustering With Adaptive Fusion

Jingxiang Chen, Quoc Tran-Dinh, Michael R. Kosorok & Yufeng Liu

To cite this article: Jingxiang Chen, Quoc Tran-Dinh, Michael R. Kosorok & Yufeng Liu (2021) Identifying Heterogeneous Effect Using Latent Supervised Clustering With Adaptive Fusion, Journal of Computational and Graphical Statistics, 30:1, 43-54, DOI: 10.1080/10618600.2020.1763808

To link to this article: https://doi.org/10.1080/10618600.2020.1763808

JOURNAL OF COMPUTATIONAL AND GRAPHICAL STATISTICS
2021, VOL. 30, NO. 1, 43–54
https://doi.org/10.1080/10618600.2020.1763808

Taylor & Francis
Taylor & Francis Group

Identifying Heterogeneous Effect Using Latent Supervised Clustering With Adaptive Fusion

Jingxiang Chen[a], Quoc Tran-Dinh[b], Michael R. Kosorok[a,b], and Yufeng Liu[a,b,c,d,e]

[a]Department of Biostatistics, The University of North Carolina at Chapel Hill, Chapel Hill, NC; [b]Department of Statistics and Operations Research, The University of North Carolina at Chapel Hill, Chapel Hill, NC; [c]Department of Genetics, The University of North Carolina at Chapel Hill, Chapel Hill, NC; [d]Carolina Center for Genome Sciences, The University of North Carolina at Chapel Hill, Chapel Hill, NC; [e]Lineberger Comprehensive Cancer Center, The University of North Carolina at Chapel Hill, Chapel Hill, NC

ABSTRACT
Precision medicine is an important area of research with the goal of identifying the optimal treatment for each individual patient. In the literature, various methods are proposed to divide the population into subgroups according to the heterogeneous effects of individuals. In this article, a new exploratory machine learning tool, named latent supervised clustering, is proposed to identify the heterogeneous subpopulations. In particular, we formulate the problem as a regression problem with subject specific coefficients, and use adaptive fusion to cluster the coefficients into subpopulations. This method has two main advantages. First, it relies on little prior knowledge and weak parametric assumptions on the underlying subpopulation structure. Second, it makes use of the outcome-predictor relationship, and hence can have competitive estimation and prediction accuracy. To estimate the parameters, we design a highly efficient accelerated proximal gradient algorithm which guarantees convergence at a competitive rate. Numerical studies show that the proposed method has competitive estimation and prediction accuracy, and can also produce interpretable clustering results for the underlying heterogeneous effect. Supplementary materials for this article are available online.

ARTICLE HISTORY
Received January 2019
Revised April 2020

KEYWORDS
Accelerated proximal gradient algorithm; Clustering analysis; Convex clustering; Machine learning; Precision medicine; Subpopulation identification

1. Introduction

In clinical research, precision medicine aims at developing the optimal treatment for each individual according to the subject's personal characteristics. The motivation originates from the findings that different groups of patients can respond dramatically differently to the same health care intervention, which can be caused by their specific body mechanisms. Failing to detect a targeted subpopulation can eliminate some precise drugs by washing out their effects using the whole population. In practice, it takes tremendous efforts to search for the targeted subpopulation of certain interventions (Brookes et al. 2004; Lagakos 2006). One important reason is that the crucial features that decide the targeted subpopulation are usually either hidden in numerous collected ones, or even unmeasured. Therefore, it is desirable to develop methods that can automatically detect such subpopulations.

Recently, various machine learning methods have been introduced and applied to identify subpopulations. In supervised learning, linear regression with two-way interactions becomes widely used. However, such a method requires strong parametric assumptions requiring that the underlying heterogeneity can be determined by those interactions (Greenland 2009). Some nonparametric methods such as random forests are also popular while the results remain less interpretable in practice (Wager and Athey 2018). In addition, there are also studies in unsupervised learning that has weak parametric assumptions. Clustering analysis can be a good representative. Clustering analysis usually detects observation similarities using a predefined distance on the covariates. One typical example is the usage of hierarchical clustering for heterogeneous gene expression analysis (Perou et al. 2000). Recently, a new clustering method, convex clustering, was proposed by solving a convex optimization problem using the pairwise fusion penalty (Guo et al. 2010; Hocking, Joulin, and Bach 2011; Chi and Lange 2015). The introduced algorithm tremendously boosts the efficiency especially when applied to large datasets. However, such unsupervised learning tools can produce meaningful and desirable results only when the subpopulations are completely determined by the covariate similarities defined. In practice, subpopulations can also heavily depend on the outcomes or even the relationship between outcomes and covariates. This can be shown by many clinical studies that aim to split the patient population according to drug effects.

Other than supervised and unsupervised learning, Wei and Kosorok (2013) recently introduced a new type of machine learning tools, named latent supervised learning, to maintain the advantage of both. Their method assumes that each observation comes with an unobserved label, that is, the latent outcome, which identifies its subpopulation and determines the underlying distribution. Furthermore, they assume that the

CONTACT Yufeng Liu yfliu@email.unc.edu Department of Statistics and Operations Research, The University of North Carolina at Chapel Hill, Chapel Hill, NC 27599.
Supplementary materials for this article are available online. Please go to www.tandfonline.com/r/JCGS.

observed outcome follows a mixture Gaussian distribution with two latent components. These latent components are assumed to be determined by a linear combination of features. There are follow-up extensions of Wei and Kosorok (2013) in the literature. For instance, Altstein and Li (2013) adopted this idea to the time-to-event response, and Shen and He (2015) suggested a logistic-normal mixture model instead. Although these methods showed competitive performance to detect the subpopulation boundaries, some drawbacks still exist. First, most of the existing methods still count on prior knowledge on the parametric functions of the subpopulation boundaries. In many exploratory studies, such information can be difficult to obtain. Second, many latent supervised learning methods rely on certain distribution assumptions of the observed outcomes as well, which may not hold in practice especially for complex subpopulation structures.

In this article, we focus on subpopulation detection, and aim to address the two drawbacks of latent supervised learning. In particular, we would like to propose an exploratory tool, named latent supervised clustering (LSC), to estimate the heterogeneous effects at the same time of clustering the samples without prior knowledge on their boundaries. To achieve these two goals, we formulate the problem as regression with subject-specific coefficients, which can be treated as the heterogeneous relationships between the outcomes and covariates from the observed data. Then we cluster such relationships with the adaptive fusion penalty, which is extended from the perspective of convex clustering. The proposed method inherits the advantages of both latent supervised learning and traditional clustering analysis. On one hand, it enables the data orient the learning process so that no assumption is needed for the subpopulation structure. On the other hand, it uses the information of both covariates and outcomes to estimate the heterogeneity so that it can both identify the subpopulation and predict the outcome values. In contrast, regular clustering does not use the outcome information.

Clustering such outcome-predictor relationships can be very challenging because they are not observed directly but can only be derived. One important question is how to define a distance properly to encourage such clustering patterns. We would like to adapt the idea of convex clustering. Note that convex clustering formulates the clustering process as a convex minimization problem involving the sum of a *loss* function and a *penalty* term with a tuning parameter balancing these two terms. The loss term is a sum of Euclidean distances between observations and their corresponding subject-specific centroids. The penalty term includes a sum of the fusion penalty between each pair of such centroids to encourage them to merge together. In LSC, we propose a different *loss* + *penalty* form to accommodate the outcome information. For the loss term, we use a predefined loss calculated from the observed outcome and its fitted value by a certain model with subject-specific parameters. In this way, a smaller loss value indicates a better goodness of fit. This model can be either parametric or nonparametric such as smoothing splines. While we only discuss linear regression in this article, extensions to classification or nonlinear regression can also be covered by, for example, using a linear function with the deviance loss or using smoothing splines with the quadratic loss. We assume that observations from the same subpopulation have the identical values of the subject-specific parameters. To encourage such a pattern in the estimation, we impose an adaptive pairwise fusion penalty on each pair of the parameters in the penalty term. The corresponding weights are determined by the estimated differences of the pairs. In summary, such a convex optimization formulation can lead to maximizing the overall goodness of fit at the same time of minimizing the heterogeneity within each detected cluster.

The main contributions of this article are as follows. First, we propose a new method using the supervised machine learning framework that aims to identify the heterogeneity by clustering the defined outcome-predictor relationship. We borrow the convex clustering idea but design new loss functions and penalty terms to encourage competitive performance and computational efficiency. Second, we design a novel optimization algorithm to solve the underlying convex minimization problem. This algorithm has several distinct features compared to existing proximal gradient methods: (1) It can handle nonsmooth loss terms using a smoothing technique as in Nesterov (2005) in a homotopy fashion; (2) It allows us to use inexact computation of the proximal operator of the penalty term; (3) As far as we know, it has the best known convergence rate guarantee among homotopy smoothing first-order methods. We would like to point out that the proposed LSC technique covers the problem that Ma and Huang (2016) discussed as a special case. They focused on the case when the subpopulations can be merely determined by a varying intercept term of a linear model.

The remainder of the article is organized as follows. In Section 2, we introduce the proposed method. In Section 3, we present the proposed accelerated proximal gradient algorithm to solve the optimization problem and show its convergence rate properties. Simulated examples and data applications are presented in Section 4 to demonstrate the performance of the proposed method under finite samples. Some discussions are provided in Section 5. More technical details including statistical learning theory and proofs of theorems, together with additional numerical results, are left in the supplementary materials.

2. Methodology

We use $\{(\boldsymbol{x}_i, y_i),\ i = 1, \ldots, n\}$ to represent our training data, where $\boldsymbol{x}_i$ is a p-dimensional covariate vector with its response y_i. We consider the model as follows:

$$y_i = f(\boldsymbol{x}_i; \boldsymbol{\beta}_i) + \varepsilon_i, \tag{1}$$

where $\boldsymbol{x}_i$ always contains an intercept term as its first element, $\boldsymbol{\beta}_i$ is the coefficient vector of the ith observation $\boldsymbol{x}_i$, and ε_i is the noise term that has mean zero and a bounded variance. We further assume that $\boldsymbol{x}_i$, and ε_i are mutually independent of each other. To describe the heterogeneity of covariate effects, we allow $\boldsymbol{\beta}_i$ to take different values for different indices i. Our goal is to estimate and cluster these $\boldsymbol{\beta}_i$'s, and let the clustering results help to provide subpopulation identification. For model (1), we assume that the value of $\boldsymbol{\beta}_i$ is determined by its underlying subpopulation. In other words, if we denote a partition of $\{1, \ldots, n\}$ with $\mathcal{S} = (\mathcal{S}_1, \ldots, \mathcal{S}_K)$, where K represents the number of subpopulations and is usually unknown in practice, then the values of $\boldsymbol{\beta}_i$ from the same latent subpopulation $\in \mathcal{S}_m$ are supposed to be identical. In this article, we concentrate on linear models, that is, $f(\boldsymbol{x}_i;\boldsymbol{\beta}_i) = \boldsymbol{x}_i^T\boldsymbol{\beta}_i$. When the linear assumption is too strong, one can extend the function to be nonlinear such as using basis expansions with $f(\boldsymbol{x}_i;\boldsymbol{\beta}_i) = \sum_{j=1}^m \boldsymbol{\beta}_{ij}g_j(\boldsymbol{x}_i)$, where $g_1,\ldots,g_m$ are basis functions.

To estimate and cluster these $\boldsymbol{\beta}_i$'s, we consider the following optimization problem:

$$\min_{\boldsymbol{\beta}_i\in\mathbb{R}^p}\left\{\phi_n(\boldsymbol{\beta}_i;\lambda_n) = \sum_{i=1}^n\left[\ell(y_i,\boldsymbol{x}_i^T\boldsymbol{\beta}_i) + \lambda_n\sum_{i<j} w_{ij}\|\boldsymbol{\beta}_i - \boldsymbol{\beta}_j\|_1\right]\right\}, \quad (2)$$

where ℓ represents a preselected loss function to measure the goodness of fit. For the loss term, we consider two popular representatives in regression: the check loss (Koenker 2005) that is used in quantile regression and the quadratic loss that is used in the least squares estimation. Note that one can easily change the loss to solve classification problems. As to the penalty term, the adaptive pairwise fusion penalty is used to adjust for the potential bias created by the ℓ_1-penalty. In practice, we find that $w_{ij} = \min\{B_w, \frac{\iota^m_{\{i,j\}}}{\|\tilde{\boldsymbol{\beta}}_i-\tilde{\boldsymbol{\beta}}_j\|_1}\}$ can be a good option, where $\iota^m_{\{i,j\}}$ indicates whether the observation j is among i's m-nearest neighbors defined by the Euclidean distance, and $\tilde{\boldsymbol{\beta}}$ is an estimate of $\boldsymbol{\beta} = (\boldsymbol{\beta}_1^T,\ldots,\boldsymbol{\beta}_n^T)^T$ which can be initialized by the local regression coefficients. The upper bound B_w is added in case some pairs of $(\tilde{\boldsymbol{\beta}}_i,\tilde{\boldsymbol{\beta}}_j)$ have values too close to each other. Considering the numerous terms of the fusion penalty ($O(n^2)$), the m-nearest neighbors strategy can save tremendous computational time when solving (2) while maintaining competitive performance.

Note that we can view our model (2) as an exact penalty formulation of the constrained problem of minimizing the loss subject to equality constraints $\boldsymbol{\beta}_i = \boldsymbol{\beta}_j$, where the product $\lambda_n w_{ij}$ can be considered as penalty parameters. From the theory of exact penalty methods, the parameters w_{ij} are expected to update at each iteration of the algorithm until they go beyond a given threshold defined by the ℓ_∞-norm of the corresponding optimal Lagrange multiplier associated with the constraint $\boldsymbol{\beta}_i = \boldsymbol{\beta}_j$ (see, e.g., Nocedal and Wright 2006, chap. 17).

In practice, one may find some prior evidence to show that certain components of $\boldsymbol{x}_i$ have homogeneous effects on the outcome y_i, that is, coefficients remain constant for all. In this case, we recommend imposing the penalty only on the heterogeneous part. Suppose that the collected observation includes $(\boldsymbol{x}_i,\boldsymbol{z}_i,y_i)$, where $\boldsymbol{z}_i$ is a q dimensional covariate vector known to have homogeneous effect. We rewrite the model (1) into $y_i = f(\boldsymbol{x}_i;\boldsymbol{\beta}_i) + h(\boldsymbol{z}_i;\boldsymbol{\gamma}) + \varepsilon_i$, where $\boldsymbol{\gamma}$ is the same for all, and h is a measurable function that can have a parametric or nonparametric form. Similar to the form of f, we restrict our discussion for linear functions with $h(\boldsymbol{z}_i;\boldsymbol{\gamma}) = \boldsymbol{z}_i^T\boldsymbol{\gamma}$. In this scenario, the optimization problem in (2) becomes

$$\min_{\boldsymbol{\beta}_i,\boldsymbol{\gamma}}\left\{\phi_n(\boldsymbol{\beta}_i,\boldsymbol{\gamma};\lambda_n) = \sum_{i=1}^n\left[\ell(y_i,\boldsymbol{x}_i^T\boldsymbol{\beta}_i + \boldsymbol{z}_i^T\boldsymbol{\gamma}) + \lambda_n\sum_{i<j} w_{ij}\|\boldsymbol{\beta}_i - \boldsymbol{\beta}_j\|_1\right]\right\}. \quad (3)$$

Moving redundant components from $\boldsymbol{x}_i$ to $\boldsymbol{z}_i$ can be crucial in saving computational time especially when the dimension of the covariates is large. Here, we suggest a "forward screening" idea that can help distinguish $\boldsymbol{z}_i$ from $\boldsymbol{x}_i$ when no prior knowledge can be obtained. We start with a parsimonious model in which $\boldsymbol{x}_i$ only contains an intercept term. Then, we move a variable from $\boldsymbol{z}_i$ to $\boldsymbol{x}_i$ that boosts the model performance the most. This process is repeated until no further improvement can be obtained by moving more variables to $\boldsymbol{x}_i$. More illustrations of this idea are left in data applications of Section 4. In the following sections, we concentrate on the model (3).

As a special case of LSC, Ma and Huang (2016) focused on when the subpopulations can be determined by a subject-specific intercept. They suggested using a concave fusion penalty in the objective function and applied the alternating direction method of multipliers (ADMM) algorithm to solve the optimization problem. However, this method is not suitable to solve our problem. To apply ADMM to (3) that has a complex regularization term, one needs to introduce many intermediate variables, which significantly increases the problem size and becomes extremely inefficient. In addition, ADMM requires a complicated strategy to tune the penalty parameter of the augmented Lagrangian to obtain good performance, which can be very difficult with complex regularization. In practice, it can be observed from Section 4 that ADMM takes longer to compute and produces suboptimal prediction accuracy.

Our proposed method and algorithm enjoy several advantages. First, our method has significant computational benefits because the problem we solve is convex, and also our proposed algorithm does not need to generate the $p \cdot n^2$ additional intermediate parameters as ADMM does. This feature allows our algorithm to scale up to relatively high dimensional problems compared to ADMM. In addition, our method has a theoretical convergence rate guarantee on the original model (3) as opposed to a guarantee on the constrained reformulation for the ADMM. Second, the quadratic loss suggested by Ma and Huang (2016) can be a suboptimal choice due to its sensitivity to the outliers. This can be partly attributed to the fact that the least square estimators have the breakdown point to be zero (Huber 2004; Zhao, Yu, and Liu 2018). That indicates, if a subject from the first subpopulation is wrongly assigned to the second subpopulation, it can highly impact the coefficient estimates of the second subpopulation when the quadratic loss is applied. We compare the results of the quadratic loss and check loss, and find the latter one can significantly improve the model performance. Third, it is not desirable to penalize all pairs of $(\boldsymbol{\beta}_i,\boldsymbol{\beta}_j)$ equally. In the ideal case, we hope that large weights can be assigned to the pairs from the same subpopulation, while zero weights are used for those from different subpopulations.

The proposed method can also be used to predict the subpopulations and response of a new observation. One can treat the identified outcome labels in the training datasets as the estimated underlying outcome, and fit it using the predictor information with a standard classification model. Some popular choices include discriminant analysis, k-nearest neighbors and random forests. The fitted classification model can be used to make predictions on which subpopulations the new observations should be assigned to. Then, one can plug in the corresponding estimated $\boldsymbol{\beta}_i$ and $\boldsymbol{\gamma}$ to predict the response y_i.

3. Algorithms

In this section, we design a novel accelerated proximal gradient algorithm to solve the proposed optimization problem efficiently, inspired by the fast iterative shrinkage-thresholding algorithm (FISTA, Beck and Teboulle 2009). To further increase the speed of convergence, we integrate a restarting strategy at each iteration, which includes inexact calculation of the proximal operator. Note that FISTA can only work for smooth objective functions. For the nonsmooth check loss, we propose approximating it with a smooth adaptive surrogate function that guarantees the best known convergence rate.

3.1. *Model Reformulation*

For simplicity, we concatenate the parameters $\boldsymbol{\beta}_i$ and $\boldsymbol{\gamma}$ of model (3) into one vector $\boldsymbol{\zeta} \triangleq (\boldsymbol{\beta}^T, \boldsymbol{\gamma}^T)^T$ where $\boldsymbol{\beta}^T = (\boldsymbol{\beta}_1^T, \ldots, \boldsymbol{\beta}_n^T)$. Then, we rewrite the loss and penalty term as

$$\begin{aligned} f_n(\boldsymbol{\zeta}) &= \sum_{i=1}^{n} \ell(y_i, \boldsymbol{x}_i^T \boldsymbol{\beta}_i + \boldsymbol{z}_i^T \gamma) \quad \text{and} \\ J_n(\boldsymbol{\zeta}) &= \lambda_n \|D_w \boldsymbol{\beta}\|_1 \\ &\equiv \lambda_n \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} \sum_{k=0}^{p} |\beta_{ik} - \beta_{jk}|, \end{aligned} \tag{4}$$

where D_w is the weight matrix. In this way, the optimization problem (3) can be expressed in the following compact form:

$$\min_{\boldsymbol{\zeta} \in \mathbb{R}^{np+q}} \left\{ \phi_n(\boldsymbol{\zeta}) = f_n(\boldsymbol{\zeta}) + J_n(\boldsymbol{\zeta}) \right\}. \tag{5}$$

We focus on two loss functions: a quadratic loss $\ell_2(r) = \frac{1}{2} r^2$ and a check loss $\ell_\tau(r) = \tau r I(r \geq 0) - (1-\tau) r I(r < 0) = (\tau - 0.5) r + 0.5|r|$. When ℓ is the quadratic loss, f_n has Lipschitz gradient, that is, there exists $L_{f_n} \geq 0$ such that $\|\nabla f_n(\boldsymbol{\zeta}) - \nabla f_n(\hat{\boldsymbol{\zeta}})\| \leq L_{f_n} \|\boldsymbol{\zeta} - \hat{\boldsymbol{\zeta}}\|$ for all $\boldsymbol{\zeta}, \hat{\boldsymbol{\zeta}}$. When ℓ is the check loss, (5) is still convex but fully nonsmooth (i.e., both f_n and J_n are nonsmooth).

3.2. *Algorithmic Design and Convergence Properties*

We develop the algorithm based on Beck and Teboulle (2009) and Nesterov (2013), while the following steps are new. First, we design a proximal operator (see the Definition) for the penalty J_n using an adaptive fast projected gradient method with a warm-start. Second, we incorporate the algorithm with a restart procedure recently studied in Fercoq and Qu (2016) to accelerate the performance of the algorithm. Third, for the check loss function, we apply smoothing techniques to approximate it with a surrogate function depending on a parameter which can be adaptively updated in the algorithm. Last, we design a new variant of the adaptive method proposed in Tran-Dinh (2017) to solve (5) that has a convergence rate guarantee without any parameter tuning strategy.

The computational cost of the algorithm consists of two parts. First, we need to evaluate the gradient vector of f_n or its smoothed approximation, and evaluate the Lipschitz constant of this gradient mapping. Second, we compute the proximal operator of J_n which is defined as

$$\text{prox}_{sJ_n}(\boldsymbol{\zeta}) := \arg\min_{\hat{\boldsymbol{\zeta}}} \left\{ J_n(\hat{\boldsymbol{\zeta}}) + \tfrac{1}{2s} \|\hat{\boldsymbol{\zeta}} - \boldsymbol{\zeta}\|_2^2 \right\}, \tag{6}$$

for any $\boldsymbol{\zeta}$ and $s > 0$. The main steps of the proposed algorithm for solving (5) are presented in Algorithm 1.

Algorithm 1 Adaptive fast proximal gradient algorithm (APG)

```
1. Choose an arbitrarily initial point $\boldsymbol{\zeta}^0 \in \mathbb{R}^{(n+1)d}$ and desired
   tolerances $\varepsilon > 0$ and $\epsilon_0 \geq 0$.
2. Evaluate $L_f := \lambda_{\max}(\tilde{X}^T \tilde{X})$; Set $\tau_0 = 1$, and $\hat{\boldsymbol{\zeta}}^0 := \boldsymbol{\zeta}^0$.
3. If the check loss is used, then input $\eta_1$ (e.g., $\eta_1 = \frac{1}{n}$).
4. For $t = 0, 1, \ldots, t_{\max}$, perform:
      Step 1: Set $L_{f_n} := L_f$ for the quadratic loss, and $L_{f_n} := \frac{L_f}{\eta_{t+1}}$
      for the check loss. Then compute the step-size $\alpha_t = \frac{1}{L_{f_n}}$.
      Step 2: Compute approximately $\boldsymbol{\zeta}^{(t+1)} \approx$
      $\text{prox}_{\alpha_t J_n}\left(\hat{\boldsymbol{\zeta}}^{(t)} - \alpha_t \nabla f_n(\hat{\boldsymbol{\zeta}}^{(t)})\right)$ up to the accuracy $\epsilon_t$
      as defined in (7).
      Step 3: If stopping-criterion is satisfied, terminate
      the algorithm.
      Step 4: If $f_n$ is the quadratic loss, update $\tau_{t+1} :=$
      $\frac{1}{2}\left(1 + \sqrt{1 + 4\tau_t^2}\right)$. If $f_n$ is the check loss, update $\tau_{t+1}$ as
      the positive solution of $\tau^3 - \tau^2 - \tau_t^2 \tau - \tau_t^2 = 0$.
      Step 5: Update the accelerated step $\hat{\boldsymbol{\zeta}}^{(t+1)} := \boldsymbol{\zeta}^{(t+1)} +$
      $\frac{\tau_k - 1}{\tau_{k+1}}\left(\boldsymbol{\zeta}^{(t+1)} - \boldsymbol{\zeta}^{(t)}\right)$.
      Step 6: If $f_n$ is the check loss, then update $\eta_{t+2} :=$
      $\left(\frac{\tau_{t+1}}{\tau_{t+1} + 1}\right)\eta_{t+1}$.
      Step 7: Perform a restarting step if needed, and update $\epsilon_{t+1}$.
5. End of the main loop.
```

In summary, when f_n is a quadratic loss, we have $f_n(\boldsymbol{\zeta}) = \frac{1}{2}\|\tilde{X}\boldsymbol{\zeta} - y\|_2^2$ where $\tilde{X} = \text{diag}(\boldsymbol{x}_1^T, \boldsymbol{x}_2^T, \ldots, \boldsymbol{x}_n^T, \boldsymbol{z}^T)$. Then, its gradient $\nabla f_n(\boldsymbol{\zeta}) = \tilde{X}^T(\tilde{X}\boldsymbol{\zeta} - y)$, which is Lipschitz continuous with its Lipschitz constant as $L_{f_n} = \lambda_{\max}(\tilde{X}^T\tilde{X})$, where $\lambda_{\max}(\tilde{X}^T\tilde{X})$ denotes the maximum eigenvalue of $\tilde{X}^T\tilde{X}$. When f_n is a nonsmooth check loss, we approximate it by a smooth $f_n(\cdot; \eta)$ with more details described in Section 3.2.2. Then, the next step is to compute the proximal operator prox_g, which is presented in Section 3.2.3.

Next we discuss the convergence guarantee of Algorithm 1. Let ζ^* be an optimal solution of (5) with its optimal value $\phi_n(\zeta^*)$, that is, we have $\phi_n(\boldsymbol{\zeta}) \geq \phi_n(\boldsymbol{\zeta}^*)$ for any $\boldsymbol{\zeta}$. Define that $\boldsymbol{\zeta}^{(t)}$ is an approximate solution to (5) with an accuracy $\varepsilon \geq 0$, if $\phi_n(\boldsymbol{\zeta}^{(t)}) - \phi_n(\boldsymbol{\zeta}^*) \leq \varepsilon$. In (5), we are not able to compute the proximal operator of J_n exactly, but rather approximate it with a given accuracy $\epsilon_t > 0$. In particular, we define

$$\begin{aligned} Q_n(\zeta; \hat{\boldsymbol{\zeta}}^{(t)}) := & f_n(\hat{\boldsymbol{\zeta}}^{(t)}) + \nabla f_n(\hat{\boldsymbol{\zeta}}^{(t)})^T(\zeta - \hat{\boldsymbol{\zeta}}^{(t)}) \\ & + \frac{L_f}{2}\|\boldsymbol{\zeta} - \hat{\boldsymbol{\zeta}}^{(t)}\|^2 + J_n(\boldsymbol{\zeta}), \end{aligned}$$

and $\boldsymbol{\zeta}_*^{(t+1)} := \text{prox}_{\alpha_t J_n}\left(\hat{\zeta}^{(t)} - \alpha_t \nabla f_n(\hat{\boldsymbol{\zeta}}^{(t)})\right)$. Then by definition, $Q_n(\cdot; \hat{\boldsymbol{\zeta}}^{(t)})$ is the corresponding objective function of the proximal operator problem at Step 2 of Algorithm 1. One can show that $\|\boldsymbol{\zeta}^{(t+1)} - \boldsymbol{\zeta}_*^{(t+1)}\| \equiv \|\boldsymbol{\zeta}^{(t+1)} - \mathrm{prox}_{\alpha_t J_n}(\hat{\boldsymbol{\zeta}}^{(t)} - \alpha_t \nabla f_n(\hat{\boldsymbol{\zeta}}^{(t)}))\| \leq \sqrt{2\alpha_t \epsilon_t}$, as long as the following condition holds:

$$0 \leq Q_n(\boldsymbol{\zeta}^{(t+1)}; \hat{\boldsymbol{\zeta}}^{(t)}) - Q_n(\boldsymbol{\zeta}_*^{(t+1)}; \hat{\boldsymbol{\zeta}}^{(t)}) \leq \epsilon_t. \tag{7}$$

Now we provide a general convergence result for Algorithm 1. For the two particular losses, we present the theorems separately in the subsections below. Their technical proofs can be found in the supplementary materials.

3.2.1. Convergence for a Quadratic Loss

If ℓ is a quadratic loss, then the following theorem provides a convergence rate guarantee of Algorithm 1.

Theorem 3.1. Let f_n be a quadratic loss, and let $\{\boldsymbol{\zeta}^{(t)}\}$ be a sequence generated by Algorithm 1 where $\mathrm{prox}_{\gamma j_n}$ is computed approximately with the accuracy $\epsilon_t \geq 0$ as defined in (7). Then, we have

$$\phi_n(\boldsymbol{\zeta}^{(t)}) - \phi_n(\boldsymbol{\zeta}^*) \leq \frac{2\lambda_{\max}(\tilde{X}^T\tilde{X})}{(t+1)^2}\left(\|\boldsymbol{\zeta}^{(0)} - \boldsymbol{\zeta}^*\| + R_t\right)^2, \tag{8}$$

where $R_t = \frac{\sqrt{2}}{\sqrt{L_f}}\left(2\sum_{j=0}^{t-1}(j+1)\sqrt{\epsilon_j} + \sqrt{\sum_{j=0}^{t-1}(j+1)^2\epsilon_j}\right)$. Consequently, for any accuracy $\varepsilon > 0$ and positive constant $c \geq 1$, if the inner accuracy at t, ϵ_t, is chosen to be $\frac{c}{(t+1)^5}$, then the maximum iteration number of (5) to achieve $\boldsymbol{\zeta}^{(t)}$ does not exceed $t_{\max} = \left\lfloor \frac{\sqrt{2\lambda_{\max}(\tilde{X}^T\tilde{X})}}{\sqrt{\varepsilon}}\|\boldsymbol{\zeta}^{(0)} - \boldsymbol{\zeta}^*\| + \frac{10\sqrt{2c}}{\sqrt{\varepsilon}}\right\rfloor$. Here, $\lfloor . \rfloor$ denotes the floor function.

3.2.2. Smoothing Technique for the Check Loss

The check loss $\rho_\tau(r) = \tau r I(r \geq 0) - (1-\tau) r I(r < 0) = (\tau - 0.5)r + 0.5|r|$ is convex but nonsmooth. We consider approximating it using a smooth convex function $\rho_\tau(\cdot;\eta)$ that has a smoothness parameter $\eta > 0$. For any fixed value of $\eta > 0$, the smooth function $\rho_\tau(\cdot;\eta)$ needs to satisfy the following basic properties. First, $\rho_\tau(\cdot;\eta)$ is smooth and convex. Its gradient $\nabla_r \rho_\tau(\cdot;\eta)$, with respect to r, is Lipschitz continuous with the Lipschitz constant L_ρ depending on η. Second, $\rho_\tau(\cdot;\eta)$ approximates $\rho_\tau(\cdot)$ well. In other words, there exists a constant D_ρ, independent of η, such that $\rho_\tau(r;\eta) \leq \rho_\tau(r) \leq \rho_\tau(r;\eta) + \eta D_\rho$ for all r. There are several choices for $\rho_\tau(\cdot)$, such as the following two:

- *Huber loss:* $\rho_\tau(r;\eta) = \begin{cases} \frac{1}{2\eta}r^2 & \text{if } |r| \leq \eta \\ |r| - \frac{\eta}{2} & \text{otherwise} \end{cases}$ with $L_\rho = \frac{1}{\eta}$ and $D_\rho = \frac{1}{2}$.
- *Logit-type loss:* $\rho_\tau(r;\eta) = \left(\tau - \frac{1}{2}\right)r + \frac{\eta}{2}\ln\left(e^{r/\eta} + e^{-r/\eta}\right)$ with $L_\rho = \frac{1}{\eta}$ and $D_\rho = \ln(2)$.

We now introduce the following properties of $\rho_\tau(\cdot;\eta)$ with the proof placed in the supplementary materials.

Lemma 3.1. We consider $f_n(\boldsymbol{\zeta};\eta) \triangleq \sum_{i=1}^n \rho_\tau(y_i(\mathbf{x}_i^T\boldsymbol{\beta}_i + \mathbf{z}_i^T\boldsymbol{\gamma});\eta)$ as a smooth version of the check loss in (4) using either the Huber loss or the logistic loss. Then, this function is convex and differentiable with its gradient $\nabla_{\boldsymbol{\zeta}} f_n(\cdot;\eta)$ as Lipschitz continuous, with the Lipschitz constant $L_{f_n} = \frac{\lambda_{\max}(\tilde{X}^T\tilde{X})}{\eta}$. Moreover, we have

$$f_n(\boldsymbol{\zeta};\eta) \leq f_n(\boldsymbol{\zeta}) \leq f_n(\boldsymbol{\zeta};\eta) + n\eta D_\rho, \tag{9}$$

for any $\boldsymbol{\zeta} \in \mathbb{R}^{(n+1)p}$ and $\eta > 0$.

The lemma shows that $f_n(\boldsymbol{\zeta};\eta) \to f_n(\boldsymbol{\zeta})$ as $\eta \to 0^+$. In this way, we approximate the problem of (5) by

$$\min_{\boldsymbol{\zeta} \in \mathbb{R}^{np+q}} \left\{\phi_n(\boldsymbol{\zeta};\eta) = f_n(\boldsymbol{\zeta};\eta) + J_n(\boldsymbol{\zeta})\right\}. \tag{10}$$

Our goal is to compute an ε-approximation solution $\boldsymbol{\zeta}^{(t)}$ to the true solution $\boldsymbol{\zeta}^*$ of (5) such that $\phi_n(\boldsymbol{\zeta}^{(t)}) - \phi_n(\boldsymbol{\zeta}^*) \leq \varepsilon$. To fulfill the goal, we propose the idea of applying the fast proximal gradient method to solve (10) approximately, with a homotopy scheme to decrease the smoothness parameter η_t at each iteration t such that $\eta_t \to 0^+$. The step-size parameter is updated by using the unique positive solution τ_{t+1} of the cubic equation $c_3(\tau) = \tau^3 - \tau^2 - \tau^{2t}\tau - \tau^{2t} = 0$.

The following theorem indicates that if $\boldsymbol{\zeta}^{(t)}$ generated by Algorithm 1 is an approximate solution of (10), then it is also an approximate solution of the original problem (5). The detailed proof is left in the supplementary materials.

Theorem 3.2. Let $\{\boldsymbol{\zeta}^{(t)}\}$ be a sequence generated by Algorithm 1, where prox_{sJ_n} is computed approximately as (7) with the accuracy $\epsilon_t := \frac{c}{(t+1)^4}$ for some positive constant $c \geq 1$. Then we have

$$\phi_n(\boldsymbol{\zeta}^{(t+1)}) - \phi_n(\boldsymbol{\zeta}^*) \leq \frac{1}{(t+1)}\left(\frac{L_f}{2\eta_1}\|\boldsymbol{\zeta}^{(0)} - \boldsymbol{\zeta}^*\|^2 \right. \tag{11}$$

$$\left. + 2n\eta_1 D_\rho + \frac{1.9\sqrt{cL_f}}{\sqrt{\eta_1}}\|\boldsymbol{\zeta}^{(0)} - \boldsymbol{\zeta}^*\| + \frac{35c}{2\eta_1} + \Gamma_t\right),$$

where $L_f := \lambda_{\max}(\tilde{X}^T\tilde{X})$, $\Gamma_t = 0$ for the Huber loss, and $\Gamma_t = \frac{D_\rho(1+4\eta_1\ln(t+1))}{2}$ for the logistic loss. Hence, for any accuracy $\varepsilon > 0$, the maximum number of iterations for an approximate solution of (5) does not exceed $t_{\max} = \mathcal{O}\left(\frac{1+\Gamma_\varepsilon}{\varepsilon}\right)$, where $\Gamma_\varepsilon = 0$ for the Huber loss, and $\Gamma_\epsilon = \ln\left(\frac{1}{\varepsilon}\right)$ for the logistic loss.

Remark 1. The worst-case convergence bounds in Theorems 3.1 and 3.2 depend on $\lambda_{\max}(\tilde{X}^T\tilde{X})$. If this eigenvalue is large, then it affects the complexity bound $t_{\max}$ of our methods. To speed up the actual performance, we suggest to perform a linesearch routine as the linesearch variant in the supplementary materials. Another possibility is to apply a preconditioning technique to $\tilde{X}^T\tilde{X}$ to reduce its leading eigenvalue before solving the problem.

3.2.3. Evaluating the Proximal Operator for the Penalty

We now describe how to approximately evaluate the proximal operator of J_n under the condition (7). We convert the problem (6) into its dual problem which is defined as

$$\mathrm{prox}_{sJ_n}(u) = u - sD_w^T v^*(u), \quad \text{where} \tag{12}$$

$$v^*(u) := \arg\min_{\|v\|_\infty \leq 1}\left\{\frac{s}{2}\|D_w^T v\|^2 - v^T D_w u\right\},$$

and D_w is defined in (4). To approximate $\nu^*(u)$, we apply an accelerated projected gradient algorithm that can be presented in a few lines as follows:

3.2.3.1. Accelerated Projected Gradient Scheme to Approximate $\mathbf{prox}_{sJ_n}(u)$. Given u, $s > 0$ and an initial point $\nu^{(0)}$. Compute $L_{D_w} := \lambda_{\max}(D_w D_w^T)$. Set $\bar{\nu}^{(0)} = \hat{\nu}^{(0)} := \nu^{(0)}$, $\delta_0 := 1$ and $\Gamma_0 := 0$. At each iteration $j \geq 0$, we update

1. $\nu^{(j+1)} := \pi_{B_\infty}\left(\hat{\nu}^{(j)} - \frac{1}{L_{D_w}} D_w(D_w^T \hat{\nu}^{(j)} - s^{-1}u)\right)$;
2. $\hat{\nu}^{(j+1)} := \nu^{(j+1)} + \frac{\delta_j - 1}{\delta_{j+1}}(\nu^{(j+1)} - \nu^{(j)})$ where $\delta_{j+1} := \frac{1}{2}(1 + \sqrt{1 + 4\delta_j^2})$;
3. $\bar{\nu}^{j+1} := (1 - \omega_j)\bar{\nu}^j + \omega_j \nu^{(j+1)}$, where $\Gamma^{j+1} := \Gamma^j + \delta_{j+1}$, and $\omega_j := \frac{\delta_{j+1}}{\Gamma_{j+1}}$.

Here, $\pi_{B_\infty}(\nu) = \max\{\min\{\nu, 1\}, -1\}$ is the projection of ν onto the ℓ_∞-unit ball $B_\infty := \{\nu \mid \|\nu\|_\infty \leq 1\}$. This algorithm is terminated after a prespecified number of iterations, $j_{\max}$. The output of this routine is $\zeta^{(t+1)} := \hat{\boldsymbol{\zeta}}^{(t)} - \alpha_t \nabla f_n(\hat{\boldsymbol{\zeta}}^{(t)}) - \alpha_t D_w^T \bar{\nu}^{j_{\max}}$, which approximates $\boldsymbol{\zeta}_*^{(t+1)} = \mathrm{prox}_{\alpha_t J_n}(\hat{\boldsymbol{\zeta}}^{(t)} - \alpha_t \nabla f_n(\hat{\boldsymbol{\zeta}}^{(t)}))$.

Next we analyze the computational effort to achieve the approximation point $\boldsymbol{\zeta}^{(t+1)}$ as in (7). By adapting the results from Tran-Dinh (2017), we have $Q_n(\boldsymbol{\zeta}^{(t+1)}; \hat{\zeta}^{(t)}) - Q_n(\boldsymbol{\zeta}_*^{(t+1)}; \hat{\zeta}^{(t)}) \leq \frac{2L_D\|\nu^{(0)} - \nu_t^*\|^2}{(j+1)^2}$, where $\nu_t^* := \nu^*(\hat{\boldsymbol{\zeta}}^{(t)} - \alpha_t \nabla f_n(\hat{\boldsymbol{\zeta}}^{(t)}))$ and the function $\nu_t^*(\cdot)$ is defined in (12). To achieve an ϵ_t-approximate solution $\boldsymbol{\zeta}^{(t+1)}$ of $\boldsymbol{\zeta}_*^{(t+1)}$ at the tth iteration, we requires $\frac{2L_{D_w}\|\nu^{(0)} - \nu_t^*\|^2}{(j+1)^2} \leq \epsilon_t$. Hence, the maximum number of iterations $j_{\max}$ is

$$j_{\max}(t) := \left\lfloor \frac{\sqrt{2\lambda_{\max}(D_w D_w^T)}}{\sqrt{\epsilon_t}} \|\nu^{(0)} - \nu_t^*\| \right\rfloor.$$

By exploiting a warm-start strategy with the previous approximate point $\nu^{(t-1)}$ for $\nu^{(0)}$, we find that the distance $\|\nu^{(0)} - \nu_t^*\|$ becomes smaller. In this way, we recommend fixing $j_{\max}$ as a small number, such as 50, to achieve an approximate $\boldsymbol{\zeta}^{(t+1)}$ in the implementation.

3.3. Finding an Initial Point

The convergence properties in Section 3.2 indicates that a good selection of the starting values $\tilde{\boldsymbol{\zeta}} = (\tilde{\boldsymbol{\beta}}, \tilde{\gamma})$ can reduce the number of iterations. Based on the assumption that each variable in X is independent of each one in Z, we introduce an ad-hoc method that can be easily applied to find a proper start point $\tilde{\boldsymbol{\beta}}$ in practice. We split the method into two steps as follows.

First, we calculate the distance matrix of the dataset based on $(X, \boldsymbol{y}^*)$, where $\boldsymbol{y}^*$ is the residual of the linear regression between $\boldsymbol{y}$ and Z. We use $\boldsymbol{y}^*$ instead of $\boldsymbol{y}$ since the expectation of $\boldsymbol{y}^*$ is exactly $X\boldsymbol{\beta}$. This conclusion holds due to the fact that a linear regression produces unbiased coefficients estimate when the omitted variables in Z are all independent of those included in the model. In this way, we treat the response $\boldsymbol{y}^*$ as a new variable and calculate the distance matrix on $(X, \boldsymbol{y}^*)$. For our numerical study, we choose to calculate the Manhattan distance, that is, $d(i,j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_1 + \alpha\|y_i^* - y_j^*\|_1$ (Borg and Groenen 2005), with $\alpha = 1$.

Second, denote $\tilde{\boldsymbol{\beta}} = (\tilde{\boldsymbol{\beta}}_1, \ldots, \tilde{\boldsymbol{\beta}}_n)$ and calculate each $\tilde{\boldsymbol{\beta}}_i$ based on the k-nearest neighbors of the ith subject with a linear regression model with the selected loss. The k-nearest neighbor set for the ith subject is defined as the k observations having the smallest distance $d(i,j)$ to the ith subject, and whose response being within the neighbor of y_i as $O_\epsilon(y_i^*)$. The reason of the second criterion is to increase the chance that those neighbors come from the same latent groups as the ith subject does. The selection of the neighbor ball radius depends on the variation of the noise and $\boldsymbol{x}_i^T\boldsymbol{\beta}_i$. In our numerical examples, we vary ϵ from 0.5 to 6.

As a remark, we note that we need to update the weights using current estimate of the parameters. To simplify the computation, instead of solving the global solution of each convex problem to update the weights, for early iterations of Algorithm 1, we use one-iteration approximation of the true solution of the convex problem, leading to a suboptimal solution to this convex problem. Although the suboptimal solution may not be very accurate, it can be used as a rough approximation to the true solution of the convex problem using the current weights by doing only one iteration of the whole optimization routine. This can greatly speed up the computation. Once the weights stabilize, our algorithm automatically converges to the true global solution using the corresponding stable weights.

4. Numerical Analysis

In this section, we use numerical examples to test the performance of LSC. We study the estimation accuracy, runtime, and prediction performance using simulated and real data. For comparisons, we also implemented the concave method by Ma and Huang (2016) with a general heterogeneous-effect vector $\boldsymbol{x}_i$ (Concave), and its extension (Concave-2, Ma, Huang, and Zhang 2018). We additionally include two standard methods: linear regression with all two-way interactions and random forests. We pick the tuning parameters by selecting λ_n from $\{2^{-3}, 2^{-2}, 2^{-1}, 2^0\}$, setting the number of neighbors as $m = 10$, and choosing the radius of the response neighbor ϵ from $\{2^{-1}, 2^0, 2, 2^2\}$.

4.1. Simulations

We evaluate the model using eight simulated examples with linear outcome-predictor relationship. In particular, Examples 1-2 include two subpopulations with linear boundaries. Example 3 is extended from the simulation in Wei and Kosorok (2013), by adding more covariates in $\boldsymbol{x}_i$. Example 4 has a higher dimensional $\boldsymbol{z}_i$ with noises. Examples 5, 6, and 8 introduce nonlinear subpopulation boundaries with noisy variables whose coefficients are zeros. Example 7 covers the nonlinear boundary case where the number of underlying subgroups increases to 5.

For each example, we generate each of the $\boldsymbol{x}_i$ and $\boldsymbol{z}_i$ from an independent continuous uniform $U(-2, 2)$, and the random noise ε_i from $N(0, 0.1)$. We use a tuning dataset and choose the λ_n that minimizes the tuning error. To calculate the validation error, we consider the estimated parameters for all detected subpopulations from the training set. We choose the one with the smallest mean square error as its validation prediction.

Once finding the optimal tuning parameters, we generate a test dataset that is ten times as large as the training to assess the prediction performance. We divide prediction into two steps. First, we treat the clustering label of the training set as an underlying outcome, and fit it using all the covariates with a classifier. In the simulation studies, we choose k-nearest neighbors for most of the cases and use kernel discriminant analysis as an alternative when $\boldsymbol{x}_i$ is high dimensional with noisy variables. Second, for each observation from the testing set, we use the fitted classifier to decide which cluster it belongs to, and then plug in the corresponding estimated coefficients to make predictions.

For the accuracy comparison, we report the averages and standard deviations mean squared errors. In addition, we also compare the estimated number of subpopulations $\hat{K}$. For the time comparison, we present the average run time (in seconds) of each method for all examples. The simulations are repeated for 50 times with the details listed below.

Example 1. Univariate linear regression with two subgroups. Suppose the underlying true model is linear with:

$$y_i = \begin{cases} 1 - x_{i1} + \boldsymbol{z}_i^T\boldsymbol{\gamma} + \varepsilon_i, & x_{i1} \le 1, \\ -1 + x_{i1} + \boldsymbol{z}_i^T\boldsymbol{\gamma} + \varepsilon_i, & x_{i1} > 1, \end{cases}$$

where $\boldsymbol{\gamma} = (1, -5, 2, 1, -3, 1, 3, 2, -4)^T$ and the random noise $\varepsilon_i \sim N(0, 0.1)$.

Example 2. Three-dimensional $\boldsymbol{x}_i$ with noisy variables in $\boldsymbol{z}_i$. The true model is

$$y_i = \begin{cases} 1 - x_{i1} - 2x_{i2} + \boldsymbol{z}_i^T\boldsymbol{\gamma} + \varepsilon_i, & x_{i1} + x_{i2} \le 0, \\ -1 + 2x_{i1} - x_{i2} + \boldsymbol{z}_i^T\boldsymbol{\gamma} + \varepsilon_i, & x_{i1} + x_{i2} > 0, \end{cases}$$

where $\boldsymbol{\gamma} = (1, -5, 3, 2, 1, 0, 0, 0, 0)^T$ and the random noise $\varepsilon_i \sim N(0, 0.1)$.

Example 3. Higher $\boldsymbol{x}_i$ dimension with noisy variables. We have $\boldsymbol{x}_i$ contain 25 variables, and the subpopulation is determined by the first five of them. The true model is

$$y_i = \begin{cases} -4 + \boldsymbol{z}_i^T\boldsymbol{\gamma} + \varepsilon_i, & 1 + x_{i1} + x_{i2} - 3x_{i3} + 2x_{i4} \le 0, \\ 1 + \boldsymbol{z}_i^T\boldsymbol{\gamma} + \varepsilon_i, & 1 + x_{i1} + x_{i2} - 3x_{i3} + 2x_{i4} > 0, \end{cases}$$

where $\boldsymbol{\gamma} = (1, -5, 3, 2, 1, 0, 0, 0, 0)^T$ and the random noise $\varepsilon_i \sim N(0, 0.1)$.

Example 4. Higher $\boldsymbol{z}_i$ dimension with three subgroups. Consider a model that has 50 homogeneous variables $\boldsymbol{z}_i$ and three latent subpopulations as

$$y_i = \begin{cases} 1 - x_{i1} + \boldsymbol{z}_i^T\boldsymbol{\gamma} + \varepsilon_i, & x_{i1} + x_{i2} + x_{i3} \le -1, \\ -1 + \boldsymbol{z}_i^T\boldsymbol{\gamma} + \varepsilon_i, & x_{i1} + x_{i2} + x_{i3} > 1, \\ 1 + x_{i1} + \boldsymbol{z}_i^T\boldsymbol{\gamma} + \varepsilon_i, & \text{otherwise}, \end{cases}$$

where $\boldsymbol{\gamma} = (1, -5, 3, 2, 1, \boldsymbol{0}_{45}^T)^T$, $\boldsymbol{0}_{45}^T$ represents a 45-dimensional zero vector and $\varepsilon_i \sim N(0, 0.1)$.

Example 5. Nonlinear subgroups boundary with noisy variables in $\boldsymbol{x}_i$. Consider a model in which x_i has 6 variables and 2 of them construct a nonlinear subpopulation boundary:

$$y_i = \begin{cases} 1 - x_{i1} - 3x_{i2} + \boldsymbol{z}_i^T\boldsymbol{\gamma} + \varepsilon_i, & x_{i1}^2 + x_{i2}^2 \le 4, \\ 1 + 3x_{i1} + x_{i2} + 4x_{i3} - x_{i5} + \boldsymbol{z}_i^T\boldsymbol{\gamma} + \varepsilon_i, & x_{i1}^2 + x_{i2}^2 > 4, \end{cases}$$

where $\boldsymbol{\gamma} = (1, -5, 3, 2, 1)^T$ and the random noise $\varepsilon_i \sim N(0, 0.1)$.

Example 6. Complex subgroups boundary with noisy variables in $\boldsymbol{x}_i$. Consider a situation that has a complex nonlinear subpopulation boundary:

$$y_i = \begin{cases} 1 + 5x_{i1} + \boldsymbol{z}_i^T\boldsymbol{\gamma} + \varepsilon_i, & x_{i1}^2 \sin x_{i2} + x_{i3}^3 + \log(x_{i4} + 5) + x_{i5} \le 5, \\ -1 - 3x_{i1} + \boldsymbol{z}_i^T\boldsymbol{\gamma} + \varepsilon_i, & x_{i1}^2 \sin x_{i2} + x_{i3}^3 + \log(x_{i4} + 5) + x_{i5} > 5, \end{cases}$$

where $\gamma = (1, -5, 3, 2, 1)$ and the random noise $\varepsilon_i \sim N(0, 0.1)$.

Example 7. Nonlinear subgroups boundaries with five subgroups. There are five subpopulations with $y_i = \sum_{k=1}^{5} I((x_{i1}^2 + x_{i2}^2) \in (b_{k-1}, b_k]) \cdot (k - 3)(1 + x_{i1} + \cdots + x_{i5}) + \boldsymbol{z}_i^T\boldsymbol{\gamma} + \varepsilon_i$, where $(b_0, b_1, b_2, b_3, b_4, b_5) = (-\infty, 2, 3.5, 5, 7, \infty)$, $\boldsymbol{\gamma} = (1, -5, 3, 2, 1)^T$, and $\varepsilon_i \sim N(0, 0.1)$.

Example 8. Complex subgroups boundaries with three subgroups and noisy variables. Consider a model having eleven x_i's and three subpopulations:

$$y_i = \begin{cases} 1 + x_{i1} + 3x_{i2} + 0x_{i3} + 3x_{i4} + 2x_{i5} + \boldsymbol{z}_i^T\boldsymbol{\gamma} + \varepsilon_i, \\ \quad x_{i1}^2 + \exp(x_{i2}) \le 2.5, \\ -1 + 3x_{i1} + x_{i2} - 2x_{i3} + 0x_{i4} - 2x_{i5} + \boldsymbol{z}_i^T\boldsymbol{\gamma} + \varepsilon_i, \\ \quad x_{i1}^2 + \exp(x_{i2}) > 5.5, \\ 1 - x_{i1} - x_{i2} - x_{i3} + 2x_{i4} - 0x_{i5} + \boldsymbol{z}_i^T\boldsymbol{\gamma} + \varepsilon_i, \\ \quad \text{o.w.}, \end{cases}$$

where $\boldsymbol{\gamma} = (1, -5, 3, 2, 1)^T$ and the random noise $\varepsilon_i \sim N(0, 0.1)$.

The simulation results are reported in Table 1. From Table 1, we can see that LSC with the check loss almost always produces the best RMSE results for $\hat{\boldsymbol{\beta}}$ and $\hat{\boldsymbol{\gamma}}$. The Concave method can maintain relatively small errors while suffers a large variability. The Concave-2 method by Ma, Huang, and Zhang (2018), which aims at estimating the subgroup boundary directly, can have a competitive estimate accuracy when the subgroup boundary is simpler as in Examples 1–4. When the underlying subpopulation structure becomes more complex, as in Examples 5 and 6, the advantage of the LSC-check becomes more clear due to its strong stability to the noise caused by being clustered in the wrong subpopulation. This demonstrates the advantage of using a more robust loss as well as the adaptive fusion penalty. When there are over two subpopulations in the setting with even more complex boundaries as in Examples 7 and 8, none of the methods can produce satisfactory estimation while the LSC-check still outperforms the others. This indicates that one should consider data cleaning or variable selection before fitting LSC in practice when the covariates have complex relationships. As to the estimates of the subpopulation numbers, the two LSC

Table 1. Simulation results: The averages and standard deviations of mean squared errors with the best results in bold.

Case	Concave			Concave-2			LSC-quad			LSC-check		
	$\hat{\beta}$	$\hat{y}$	$\hat{K}$	$\hat{\beta}$	$\hat{y}$	$\hat{K}$	$\hat{\beta}$	$\hat{y}$	$\hat{K}$	$\hat{\beta}$	$\hat{y}$	$\hat{K}$
1	0.261 (0.153)	0.034 (0.054)	3.04 (0.68)	0.214 (0.052)	**0.010** (0.004)	2.02 (0.14)	0.195 (0.061)	0.011 (0.008)	2.1 (0.48)	**0.185** (0.033)	0.010 (0.006)	2.17 (0.37)
2	0.349 (0.116)	0.123 (0.014)	3.15 (0.61)	0.236 (0.035)	0.017 (0.005)	2.46 (0.50)	0.282 (0.133)	0.012 (0.003)	2.27 (0.52)	**0.225** (0.092)	**0.006** (0.002)	2.21 (0.45)
3	0.408 (0.22)	0.115 (0.04)	2.32 (0.68)	**0.111** (0.002)	**0.016** (0.007)	1.87 (0.42)	0.315 (0.180)	0.032 (0.024)	2.21 (0.49)	0.177 (0.122)	0.023 (0.018)	2.11 (0.41)
4	0.421 (0.035)	0.099 (0.012)	2.67 (0.89)	0.381 (0.12)	0.012 (0.003)	2.62 (0.32)	0.285 (0.041)	**0.010** (0.004)	3.34 (0.73)	**0.273** (0.03)	0.012 (0.007)	3.12 (0.38)
5	0.434 (0.058)	0.254 (0.140)	2.32 (0.36)	0.359 (0.104)	0.076 (0.040)	2.61 (0.84)	0.332 (0.070)	0.009 (0.001)	2.14 (0.36)	**0.285** (0.041)	**0.006** (0.001)	2.1 (0.27)
6	0.616 (0.161)	0.091 (0.023)	2.73 (0.52)	0.654 (0.274)	0.293 (0.189)	2.89 (0.92)	**0.348** (0.152)	0.078 (0.001)	2.15 (0.22)	0.386 (0.134)	**0.040** (0.001)	2.13 (0.18)
7	2.345 (0.251)	0.148 (0.011)	6.25 (1.54)	2.129 (0.047)	0.083 (0.010)	6.40 (0.89)	**1.699** (0.271)	0.111 (0.098)	6.17 (1.06)	1.896 (0.157)	**0.084** (0.044)	6.23 (0.36)
8	2.134 (0.360)	0.230 (0.184)	5.53 (0.82)	2.159 (0.062)	0.257 (0.038)	4.67 (0.89)	1.349 (0.236)	0.219 (0.072)	4.47 (0.70)	**1.341** (0.262)	**0.129** (0.032)	4.32 (0.59)

NOTE: The *K* column provides the detected numbers of clusters. Concave is the method with the concave fusion penalty, Concave-2 refers to Ma, Huang, and Zhang (2018), LSC-quad and LSC-check represent latent supervised clustering with the quadratic and check loss.

Table 2. Simulation runtime comparison: average runtime (in seconds) of the selected methods for each set of the tuning parameters with the best results in bold.

Examples								
#	n	p	q	K	Concave	Concave-2	LSC-quad	LSC-check
1	300	2	9	2	18.5	15.8	**11.0**	12.1
2	300	3	9	2	692	91.7	**71.1**	107.3
3	300	24	5	2	16,954	**1054.1**	2364.1	2433.7
4	300	4	50	3	793.8	771.7	**161.5**	186.5
5	300	6	5	2	2225.1	1793.5	**458.5**	537.1
6	300	6	5	2	2244.5	6320.7	**436.7**	507.4
7	600	6	5	5	9513	7078.6	**2666.6**	3373.1
8	600	11	5	3	40,514.5	35,540.8	**11,116.2**	14,739.0

Table 3. Simulation prediction accuracy: the mean squared prediction errors and standard deviations of the selected methods with the best results in bold.

Example	Concave	LSC-quad	LSC-check	Reg-Int	RF
1	0.284 (0.05)	0.083 (0.004)	**0.057** (0.003)	0.388 (0.025)	0.499 (0.081)
2	1.142 (0.155)	0.644 (0.093)	**0.486** (0.085)	2.897 (0.13)	1.513 (0.126)
3	10.531 (0.308)	6.936 (0.202)	**3.481** (0.166)	3.599 (0.113)	9.759 (1.505)
4	2.838 (0.243)	0.935 (0.105)	**0.776** (0.106)	1.506 (0.074)	7.469 (1.21)
5	12.453 (1.893)	10.379 (1.778)	**9.985** (1.759)	30.325 (1.017)	46.881 (2.835)
6	14.176 (2.073)	11.226 (1.295)	**10.305** (1.26)	122.594 (12.372)	24.77 (3.999)
7	14.639 (1.954)	12.379 (0.91)	**9.75** (0.851)	20.829 (0.675)	16.787 (2.041)
8	32.274 (2.797)	21.748 (1.495)	**16.92** (1.351)	67.239 (1.397)	42.693 (2.894)

NOTE: Reg-Int means linear regression with two-way interactions of x_i's, and RF represents random forests.

methods perform better than the Concave while all methods tend to overestimate this number except for the case of $K = 5$ for Example 7.

Table 2 reports the average runtime that the selected methods take to return the results with a given set of tuning parameters. Latent supervised learning is faster than the concave penalty method with ADMM when the sample size n and dimension p are large. This is because the concave penalty method takes more iterations before convergence, and also ADMM creates $O(n^2p)$ intermediate parameters. Concave-2 is the fastest in Example 3 because the boundary is quite easy to be estimated directly so that their algorithm can converge much faster. In addition, the suggested specification of the weight vector $\boldsymbol{w}$ in LSC results in a sparse penalty coefficient matrix. For the two LSC methods, LSC-quad performs faster than LSC-check because the smooth check loss decreases the convergence speed of the proposed algorithm, as shown in the previous section.

To make predictions for the test datasets, we choose k-nearest neighbors with $k = 10$ to predict the underlying label except for Examples 3, 7, and 8 due to the noisy covariates. As an alternative, we choose kernel discriminant analysis with the parameters selected by the tuning sets.

We report the mean squared errors of prediction in Table 3. From the results, LSC with the check loss enjoys the minimal prediction error. For Examples 1 and 2, all the methods produce satisfactory results. Linear regression with two-way interactions performs as good as LSC in Example 3 because the underlying boundary fits the interaction assumption exactly. When the underlying boundaries become nonlinear as for Examples 5–8, neither linear regression with interactions nor random forests can produce reliable prediction results. The performance of LSC is significantly better than the concave penalty methods in terms of the average prediction error and the variability. Similar to the estimation accuracy results, none of these methods can produce good prediction results when the latent subpopulation structure becomes too complicate as in Examples 7 and 8.

To better illustrate how the value of λ_n affects the number of detected clusters, we draw the solution paths for the cases with both quadratic and check loss functions of Example 1 in Figure 1. The solution path is for the heterogeneous effect coefficients with the same idea as in the dendrogram for agglomerative hierarchical clustering (Hastie, Tibshirani, and Friedman 2009). It is calculated by using a "bottom up" approach starting with a small λ_n that does not show any clustering pattern. In

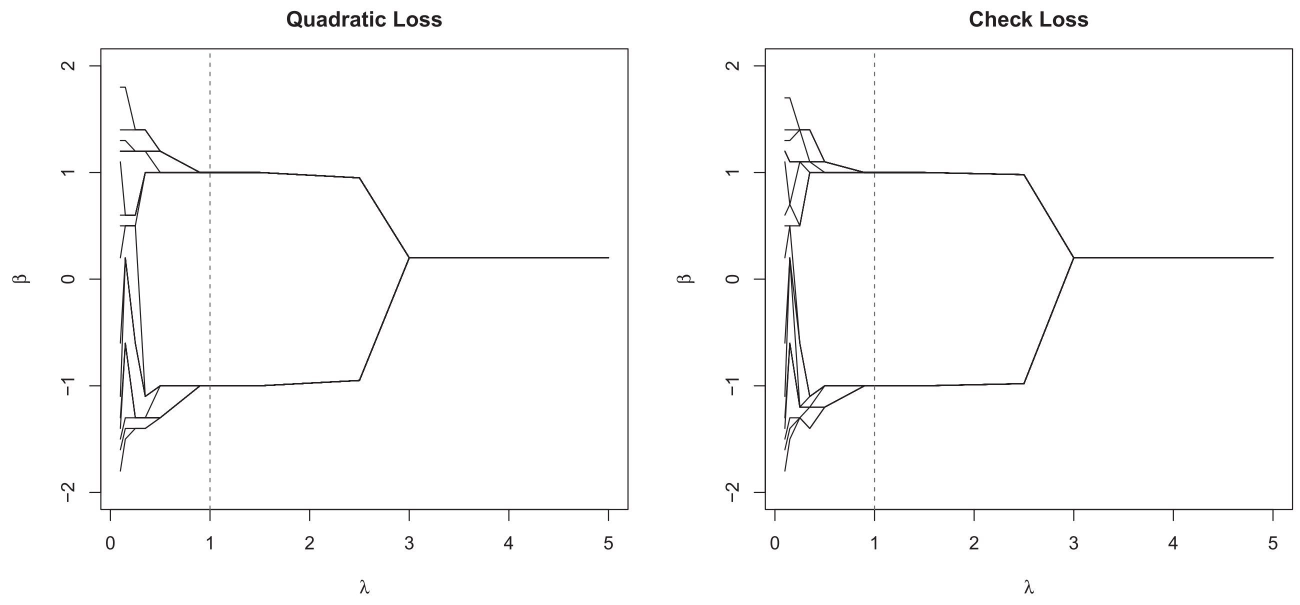

Figure 1. Solution paths for $\hat{\beta}_n$ against λ_n in Example 1 with selected λ_n shown by dashed lines.

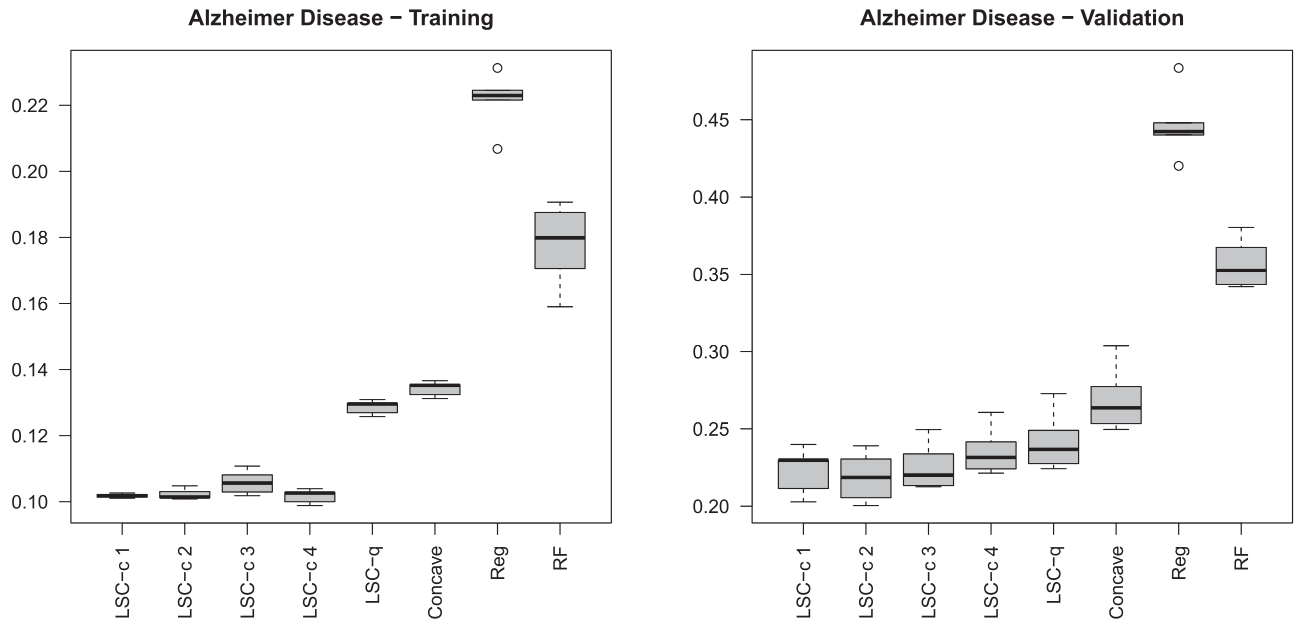

Figure 2. Alzheimer's disease: Mean squared prediction errors. LSC-c1—LSC-c4 present latent supervised clustering with the check loss that includes the corresponding number of ROI in heterogeneous vector by the "forward screening" idea, LSC-q is LSC with the quadratic loss, Concave represents the method with concave fusion penalty, Reg means linear regression with two-way interactions, and RF represents random forests.

Figure 1, the solution paths for the two loss functions look similar to each other. When λ_n reaches to around 1.5, the coefficient estimates merge into two distinct values around −1 and 1, that is, the underlying true values. As λ_n goes to a larger value such as 4, the estimates eventually converge to a single point that is close to zero.

Following a reviewer's suggestion, we have performed additional simulation studies to investigate the performance of our "forward screening" idea and the effect of correlations among predictors. The results show that our proposed forward screening is effective in identifying variable roles. When the predictors are correlated, our algorithms take more iterations to converge as the correlation increases. This matches with our theoretical analysis. More details can be found in the supplementary materials.

4.2. Real Data Applications

In this subsection, we apply the proposed method to the Alzheimer's Disease Neuroimaging Initiative (ADNI) data (Kueper, Speechley, and Montero-Odasso 2018) and Pima Indian Diabetes data (Dua and Graff 2019) to evaluate its performance. We use the same tuning parameter ranges as the simulation examples. We use a 5-fold cross-validation with 50 replications for tuning and prediction. To predict the cluster labels of observations in the validation sets, we use k-nearest neighbors with $k = 10$. For both the proposed methods and the method with the concave penalty, we follow the suggested "forward screening" idea in Section 2 to identify $\boldsymbol{x}_i$ from $\boldsymbol{z}_i$. In particular, we start with a parsimonious model that has only an intercept term in $\boldsymbol{x}_i$ and all the covariates in $\boldsymbol{z}_i$. Then we

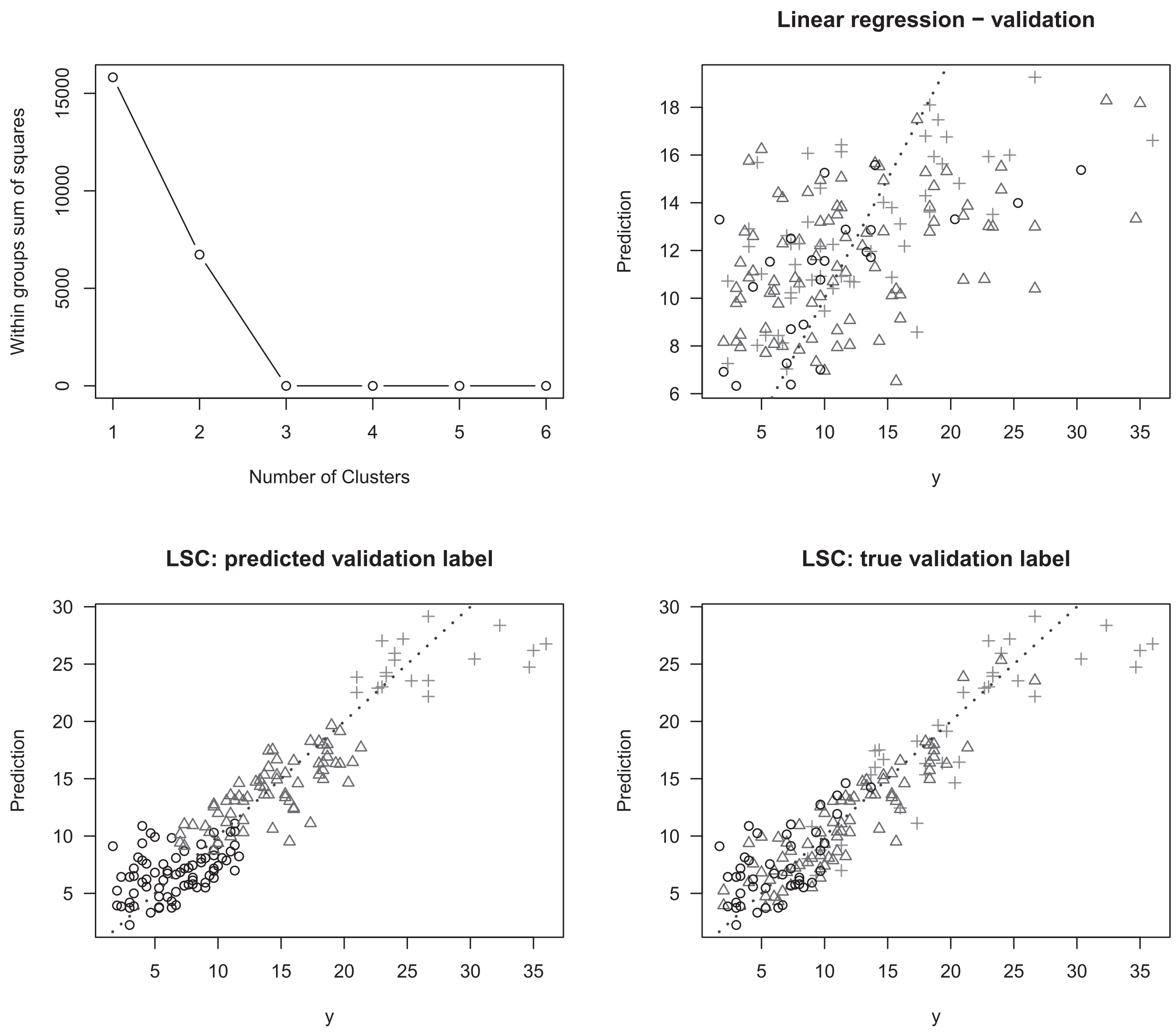

Figure 3. Alzheimer's disease: Within-cluster variation plot and scatterplots of the observed and predicted outcome for the validation set in one realization of CV.

move variables from $\boldsymbol{z}_i$ to $\boldsymbol{x}_i$ one at a time, by choosing the one that boosts the validation prediction accuracy the most. The process stops when the validation prediction is not improved. To save time in practice, one may conduct the screening with a fixed and reasonable tuning parameter set. We report the mean squared prediction errors of all the methods for both the training and validation sets as a criterion, and then briefly describe the pattern of the detected subpopulations.

4.2.1. Alzheimer's Disease Neuroimaging Initiative (ADNI)

The dataset records the structural magnetic resonance imaging (MRI) of 226 normal controls, 393 patients with mild cognitive impairment (MCI), and 186 patients with Alzheimer's disease. It uses a score (ranged from 0 to 150) named Alzheimer's Disease Assessment Scale—Cognitive Subscale (ADAS-Cog) to assess the level of cognitive dysfunction in Alzheimer's disease. The goal is to use 93 manually labeled regions of interest (ROI) as covariates, to predict the ADAS-Cog and also detect the three subgroups simultaneously (i.e., normal, MCI, and AD).

We first reduce the covariate dimension to 20 by recurrent feature elimination (Hastie, Tibshirani, and Friedman 2009), and sort the remaining covariates using the "forward screening" idea. The indices of the first five ROI are 42, 82, 43, 85, and 30. Fitting regression by subgroups, we find the coefficients of these five ROIs quite different across the three subgroups. In this way, it is reasonable to assume that the ROIs have heterogeneous effect on ADAS. Figure 2 shows that the best prediction accuracy is achieved when the ROI 42 and 82 are included. The columns of LSC-quad and Concave represent LSC results with the quadratic loss and the method with the concave fusion penalty. Both of them perform worse than LSC-check with larger prediction errors and variability. Reg and RF denote linear regression with two-way interactions and random forests. These two commonly used methods fail to achieve satisfactory prediction results for this problem.

We are also interested in comparing the detected subgroups with the true diagnosed label. Figure 3 presents the K-means clustering results of the estimated β's, the validation performance of linear regression and LSC with predicted subgroups as well as true subgroup labels. One can clearly see that the K-means suggests the correct number of clusters according to the within group sum of squares. The detected subgroups match well with the ground truth (around 60% accuracy) except for

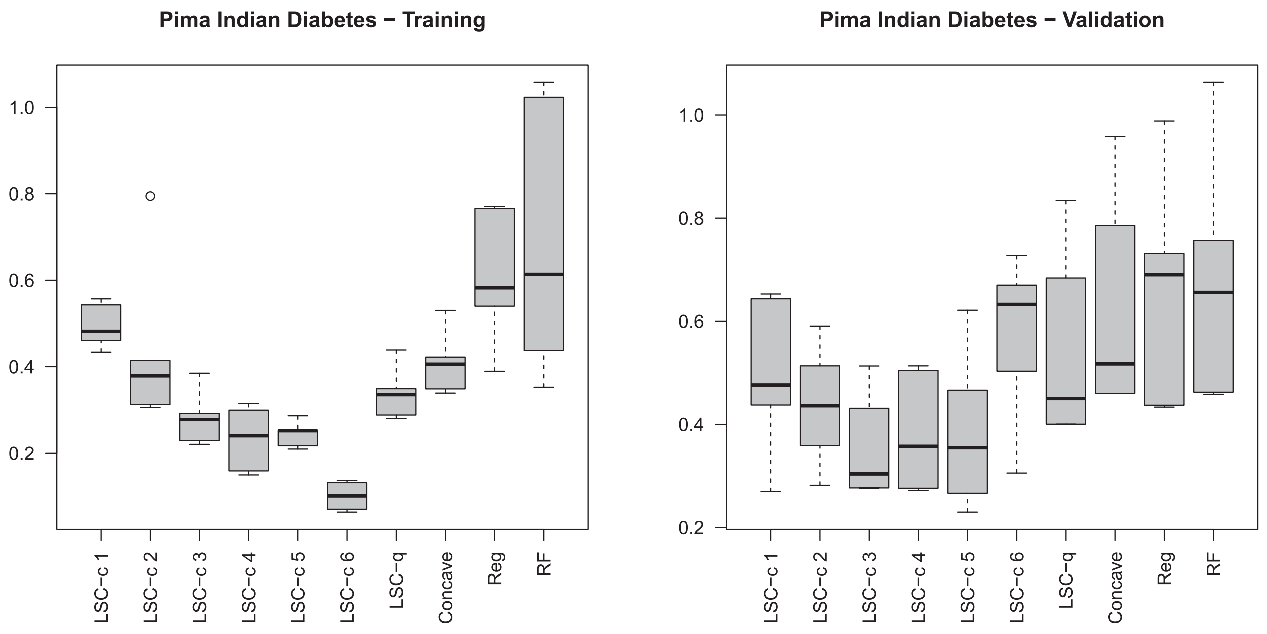

Figure 4. Pima Indian Diabetes: Mean squared prediction errors. LSC-c1–LSC-c6 present latent supervised clustering with the check loss that includes the corresponding number of variables in $\boldsymbol{x}_i$ by the "forward screening" idea.

those lie around the boundary between normal and MCI, which is proved to be difficult in the literature.

4.2.2. *Pima Indian Diabetes*

The Pima Indian Diabetes dataset collects 768 females at least 21 years old of Pima Indian heritage. The dataset contains 8 attributes and a class variable indicating whether tested positive for diabetes. The 2-hr serum insulin is measured among the 8 attributes, and it is considered as a proper surrogate outcome for the binary indicator of the underlying diabetes test. Therefore, we fit the 2-hr serum insulin using all the other attributes except for the diabetes test binary variable. We remove all the rows that contains missing values. Similar to the previous example, we use a 5-fold cross-validation and fit the selected methods using the training sets. We also apply the "forward screening" idea and the selection order is diabetes pedigree function (1), diastolic blood pressure (2), body mass index (3), age (4), triceps skin fold thickness (5), and plasma glucose concentration (6). The mean squared prediction errors are presented in Figure 4 for all the methods. The proposed method with the check loss and three variables in $\boldsymbol{x}_i$ achieves the best prediction performance. LSC-c4 and LSC-c5 can have competitive mean squared errors while the variances are slightly larger. LSC-c6 suffers overfiting with its prediction error larger than that of LSC with the quadratic loss. The linear regression with interactions and random forests produce the worst prediction errors when compared with other methods. In addition, because there is no ground truth for the subgroup in the data, we conduct a χ^2 test between the detected subgroup labels and the underlying diagnosis variable to roughly evaluate how important the labels are. The result shows that the proposed methods always suggested two latent subgroups, and the detected labels show significant relationship with the underlying diabetes test indicator according to the χ^2 test. The median of the p-values is 0.031 and this suggests that the identified subgroup is reasonable.

5. Discussion

In this article, we propose a novel machine learning method that aims at clustering the underlying subpopulation structure based on the heterogeneous relationship between the outcome and covariates. Although we mainly focus on the scenario of the linear relationship between the outcome and covariates, the proposed method can be a very good exploratory tool in practice due to its weak assumptions on the underlying subpopulation structure. We develop a very efficient algorithm with competitive convergence rates to solve the optimization problem, and also discuss the statistical consistency properties of the estimators for the coefficients. In numerical studies, the proposed method demonstrates strong performance in both subpopulation detection and outcome prediction. One interesting future direction is to extend the proposed method to other clustering methods with various types of outcomes.

Supplementary Materials

The supplementary materials include statistical learning theory of the proposed methods, technical proofs and discussions, and additional details for the numerical examples.

Acknowledgments

The authors are indebted to the editor, the associate editor, and two reviewers, whose helpful suggestions led to a much improved presentation.

Funding

This research was supported in part by NSF grants IIS1632951 and DMS1821231, NIH grants R01GM126550 and P01CA142538, and Office of Naval Research (ONR), grant no. N00014-20-1-2088.

References

Altstein, L., and Li, G. (2013), "Latent Subgroup Analysis of a Randomized Clinical Trial Through a Semiparametric Accelerated Failure Time Mixture Model," *Biometrics*, 69, 52–61. [44]

Beck, A., and Teboulle, M. (2009), "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems," *SIAM Journal on Imaging Sciences*, 2, 183–202. [46]

Borg, I., and Groenen, P. J. F. (2005), *Modern Multidimensional Scaling: Theory and Applications*, New York: Springer. [48]

Brookes, S. T., Whitely, E., Egger, M., Smith, G. D., Mulheran, P. A., and Peters, T. J. (2004), "Subgroup Analyses in Randomized Trials: Risks of Subgroup-Specific Analyses; Power and Sample Size for the Interaction Test," *Journal of Clinical Epidemiology*, 57, 229–236. [43]

Chi, E. C., and Lange, K. (2015), "Splitting Methods for Convex Clustering," *Journal of Computational and Graphical Statistics*, 24, 994–1013. [43]

Dua, D., and Graff, C. (2019), UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science. *http://archive.ics.uci.edu/ml* [51]

Fercoq, O., and Qu, Z. (2016), "Restarting Accelerated Gradient Methods With a Rough Strong Convexity Estimate," arXiv no. 1609.07358. [46]

Greenland, S. (2009), "Interactions in Epidemiology: Relevance, Identification, and Estimation," *Epidemiology*, 20, 14–17. [43]

Guo, J., Levina, E., Michailidis, G., and Zhu, J. (2010), "Pairwise Variable Selection for High-Dimensional Model-Based Clustering," *Biometrics*, 66, 793–804. [43]

Hastie, T., Tibshirani, R., and Friedman, J. (2009), *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.), New York: Springer-Verlag. [50,52]

Hocking, T. D., Joulin, A., and Bach, F. (2011), "Clusterpath: An Algorithm for Clustering Using Convex Fusion Penalties," in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA. [43]

Huber, P. J. (2004), *Robust Statistics*, New York: Wiley. [45]

Koenker, R. (2005), *Quantile Regression*, New York: Cambridge University Press. [45]

Kueper, J. K., Speechley, M., and Montero-Odasso, M. (2018), "The Alzheimer's Disease Assessment Scale—Cognitive Subscale (ADAS-COG): Modifications and Responsiveness in Pre-Dementia Populations. A Narrative Review," *Journal of Alzheimer's Disease*, 63, 423–444. [51]

Lagakos, S. W. (2006), "The Challenge of Subgroup Analyses—Reporting Without Distorting," *The New England Journal of Medicine*, 354, 1667–1669. [43]

Ma, S., and Huang, J. (2016), "A Concave Pairwise Fusion Approach to Subgroup Analysis," *Journal of the American Statistical Association*, 112, 1–42. [44,45,48]

Ma, S., Huang, J., and Zhang, Z. (2018), "Exploration of Heterogeneous Treatment Effects via Concave Fusion," arXiv no. 1607.03717. [48,49,50]

Nesterov, Y. (2005), "Smooth Minimization of Non-Smooth Functions," *Mathematical Programming*, 103, 127–152. [44]

——— (2013), "Gradient Methods for Minimizing Composite Objective Function," *Mathematical Programming*, 140, 125–161. [46]

Nocedal, J., and Wright, S. (2006), *Numerical Optimization*, New York: Springer. [45]

Perou, C. M., Sørlie, T., Eisen, M. B., van de Rijn, M., Jeffrey, S. S., Rees, C. A., Pollack, J. R., Ross, D. T., Johnsen, H., Akslen, L. A., Fluge, O., Pergamenschikov, A., Williams, C., Zhu, S. X., Lønning, P. E., Børresen-Dale, A. L., Brown, P. O., and Botstein, D. (2000), "Molecular Portraits of Human Breast Tumours," *Nature*, 406, 747–752. [43]

Shen, J., and He, X. (2015), "Inference for Subgroup Analysis With a Structured Logistic-Normal Mixture Model," *Journal of the American Statistical Association*, 110, 303–312. [44]

Tran-Dinh, Q. (2017), "Adaptive Smoothing Algorithms for Nonsmooth Composite Convex Minimization," *Computational Optimization and Applications*, 66, 425–451. [46,48]

Wager, S., and Athey, S. (2018), "Estimation and Inference of Heterogeneous Treatment Effects Using Random Forests," *Journal of the American Statistical Association*, 113, 1228–1242. [43]

Wei, S., and Kosorok, M. R. (2013), "Latent Supervised Learning," *Journal of the American Statistical Association*, 108, 957–970. [43,44,48]

Zhao, J., Yu, G., and Liu, Y. (2018), "Angle Breakdown Point for Classification," *Annals of Statistics*, 46, 3362–3389. [45]