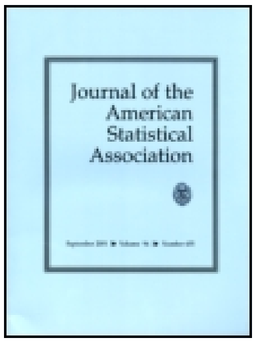

Taylor & Francis
Taylor & Francis Group

Journal of the American Statistical Association

ISSN: 0162-1459 (Print) 1537-274X (Online) Journal homepage: https://www.tandfonline.com/loi/uasa20

# Rare Feature Selection in High Dimensions

Xiaohan Yan & Jacob Bien

**To cite this article:** Xiaohan Yan & Jacob Bien (2020): Rare Feature Selection in High Dimensions, Journal of the American Statistical Association, DOI: 10.1080/01621459.2020.1796677

**To link to this article:** https://doi.org/10.1080/01621459.2020.1796677

View supplementary material

Published online: 01 Sep 2020.

Submit your article to this journal

Article views: 313

View related articles

View Crossmark data

Full Terms & Conditions of access and use can be found at
https://www.tandfonline.com/action/journalInformation?journalCode=uasa20

JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION
2020, VOL. 00, NO. 0, 1–14: Theory and Methods
https://doi.org/10.1080/01621459.2020.1796677

# Rare Feature Selection in High Dimensions

Xiaohan Yan[a] and Jacob Bien[b]

[a]Microsoft Azure, Redmond, WA; [b]Data Sciences and Operations, USC Marshall, Los Angeles, CA

**ABSTRACT**
It is common in modern prediction problems for many predictor variables to be counts of rarely occurring events. This leads to design matrices in which many columns are highly sparse. The challenge posed by such "rare features" has received little attention despite its prevalence in diverse areas, ranging from natural language processing (e.g., rare words) to biology (e.g., rare species). We show, both theoretically and empirically, that not explicitly accounting for the rareness of features can greatly reduce the effectiveness of an analysis. We next propose a framework for aggregating rare features into denser features in a flexible manner that creates better predictors of the response. Our strategy leverages side information in the form of a tree that encodes feature similarity. We apply our method to data from TripAdvisor, in which we predict the numerical rating of a hotel based on the text of the associated review. Our method achieves high accuracy by making effective use of rare words; by contrast, the lasso is unable to identify highly predictive words if they are too rare. A companion R package, called `rare`, implements our new estimator, using the alternating direction method of multipliers. Supplementary materials for this article are available online.

**ARTICLE HISTORY**
Received March 2018
Accepted July 2020

**KEYWORDS**
ADMM; Feature aggregation; Sparsity in data; Text data

## 1. Introduction

The assumption of parameter sparsity plays an important simplifying role in high-dimensional statistics. However, this article is focused on sparsity in the data itself, which actually makes estimation more challenging. In many modern prediction problems, the design matrix has many columns that are highly sparse. This arises when the features record the frequency of events (or the number of times certain properties hold). While a small number of these events may be common, there is typically a very large number of rare events, which correspond to features that are zero for nearly all observations. We call these predictors *rare features*. Rare features are in fact extremely common in many modern datasets. For example, consider the task of predicting user behavior based on past website visits: only a small number of sites are visited by a lot of the users; all other sites are visited by only a small proportion of users. As another example, consider text mining, in which one makes predictions about documents based on the terms used. A typical approach is to create a document-term matrix in which each column encodes a term's frequency across documents. In such domains, it is often the case that the majority of the terms appear very infrequently across the documents; hence, the corresponding columns in the document-term matrix are very sparse (e.g., Forman 2003; Huang 2008; Liu et al. 2010; Wang, Lu, and Zhai 2010). In Section 6, we study a text dataset with more than 200,000 reviews crawled from *https://www.tripadvisor.com*. Our goal is to use the adjectives in a review to predict a user's numerical rating of a hotel. As shown in the right panel of Figure 7, the distribution of adjective density, defined as the proportion of documents containing an adjective, is extremely right-skewed, with many adjectives occurring very infrequently in the corpus. In fact, we find that more than 95% of the 7787 adjectives appear in less than 5% of the reviews. It is common practice to simply discard rare terms,[1] which may mean removing most of the terms (e.g., Forman 2003; Huang 2008; Liu et al. 2010; Wang, Lu, and Zhai 2010).

Rare features also arise in various scientific fields. For example, microbiome data measure the abundances of a large number of microbial species in a given environment. Researchers use next generation sequencing technologies, clustering these reads into "operational taxonomic units" (OTUs), which are roughly thought of as different species of microbe (e.g., Schloss et al. 2009; Caporaso et al. 2010). In practice, many OTUs are rare, and researchers often aggregate the OTUs to genus or higher levels (e.g., Zhang, Shao, and Ye 2012; Chen et al. 2013; Xia et al. 2013; Lin et al. 2014; Randolph et al. 2015; Shi, Zhang, and Li 2016; Cao, Zhang, and Li 2017) or with unsupervised clustering techniques (e.g., McMurdie and Holmes 2013; Wang and Zhao 2017b) to create denser features. However, even after this step, a large portion of these aggregated OTUs are still found to be too sparse and thus are discarded (e.g., Zhang, Shao, and Ye 2012; Chen et al. 2013; Shi, Zhang, and Li 2016; Wang and Zhao 2017b). The rationale for this elimination of rare OTUs is that there needs to be enough variation among samples for an OTU to be successfully estimated in a statistical model (Ridenhour et al. 2017).

**CONTACT** Xiaohan Yan *xy257@cornell.edu* One Microsoft Way, Redmond, WA 98052.
Color versions of one or more of the figures in the article can be found online at *www.tandfonline.com/r/JASA*.
Supplementary materials for this article are available online. Please go to *www.tandfonline.com/r/JASA*.

[1]For example, in the `R` text mining library `tm` (Feinerer and Hornik 2017), `removeSparseTerms` is a commonly used function for removing any terms with sparsity level above a certain threshold.

© 2020 American Statistical Association

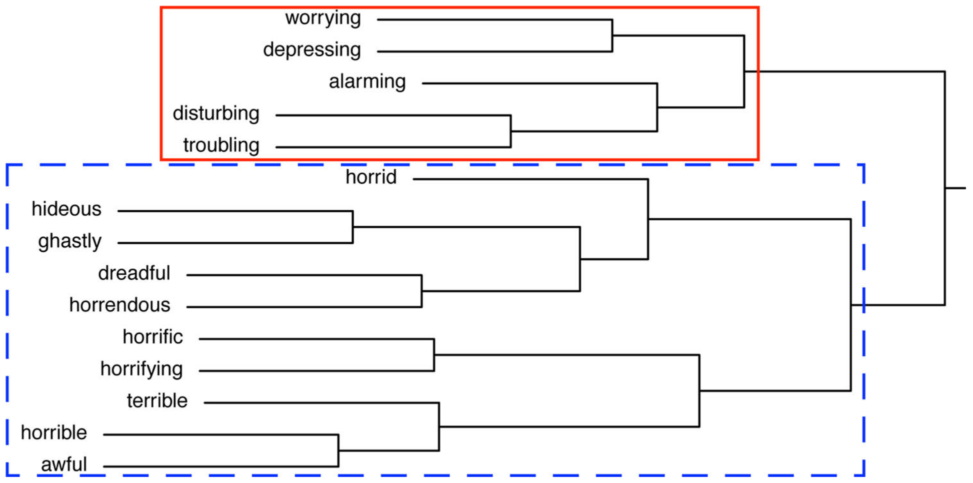

**Figure 1.** A tree that relates adjectives on its leaves.

The practice of discarding rare features is wasteful: a rare feature should not be interpreted as an unimportant one since it can be highly predictive of the response. For instance, using the word "ghastly" in a hotel review delivers an obvious negative sentiment, but this adjective appears very infrequently in TripAdvisor reviews. Discarding an informative word like "ghastly" simply because it is rare clearly seems inadvisable. To throw out over half of one's features is to ignore what may be a huge amount of useful information.

Even if rare features are not explicitly discarded, many existing variable selection methods are unable to select them. The challenge is that with limited examples there is very little information to identify a rare feature as important. Theorem 1 shows that even a single rare feature can render ordinary least squares (OLS) inconsistent in the classical limit of infinite sample size and fixed dimension.

To address the challenge posed by rare features, we propose in this work a method for forming new aggregated features which are less sparse than the original ones and may be more relevant to the prediction task. Consider the following features, which represent the frequency of certain adjectives used in hotel reviews:

- $X_{\text{worrying}}, X_{\text{depressing}}, \ldots, X_{\text{troubling}}$,
- $X_{\text{horrid}}, X_{\text{hideous}}, \ldots, X_{\text{awful}}$.

While both sets of adjectives express negative sentiments, the first set (which might be summarized as "worry") seems more mild than the second set (which might be summarized as "horrification"). In predicting the rating of a hotel review, we might find the following two aggregated features more relevant:

$$\widetilde{X}_{\text{worry}} = X_{\text{worrying}} + X_{\text{depressing}} + \cdots + X_{\text{troubling}},$$
$$\widetilde{X}_{\text{horrification}} = X_{\text{horrid}} + X_{\text{hideous}} + \cdots + X_{\text{awful}}.$$

The distinction between "horrid" and "hideous" might not matter for predicting the hotel rating, whereas the distinction between a "worry"-related word versus a "horrification"-related word may be quite relevant. Thus, not only are these aggregated features less rare than the original features, but they may also be more relevant to the prediction task. A method that selects the aggregated feature $\widetilde{X}_{\text{horrification}}$ thereby can incorporate the information conveyed in the use of "hideous" into the prediction task; this same method may be unable to otherwise determine the effect of "hideous" by itself since it is too rare.

Indeed, appropriate aggregation of rare features in certain situations can be key to attaining consistent estimation and support recovery. In Theorem 2, we consider a setting where all features are rare and a natural aggregation rule exists among the features. In that setting, we show that the lasso (Tibshirani 1996) fails to attain high-probability support recovery (for all values of its tuning parameter), whereas an oracle-aggregator does attain this property. Theorem 2 demonstrates the value of proper aggregation for accurate feature selection when features are rare. This motivates the remainder of the article, in which we devise a strategy for determining an effective feature aggregation based on data. Our aggregation procedure makes use of side information about the features, which we find is available in many domains. In particular, we assume that a tree is available that represents the closeness of features. For example, Figure 1 shows a tree for the previous word example that is generated via hierarchical clustering over *GloVe* (Pennington, Socher, and Manning 2014) embeddings learned from a different data source. The two contours enclose two subtrees resulting from a cut at their joint node. Aggregating the counts in these subtrees leads to the new features $\widetilde{X}_{\text{worry}}$ and $\widetilde{X}_{\text{horrification}}$ described above. We give more details of constructing such a tree in Section 3.1.

In Section 2, we motivate our work by providing theoretical results demonstrating the difficulty that OLS and the lasso have with rare features. We further show that correct aggregation of rare features leads to signed support recovery in a setting where the lasso is unable to attain this property. In Section 3, we introduce a tree-based parameterization strategy that translates the feature aggregation problem to a sparse modeling problem. Our main proposal is an estimator formulated as a solution to a convex optimization problem for which we derive an efficient algorithm. We draw connections between this method and related approaches and then in Section 4, provide a bound on the prediction error for our method. Finally, we demonstrate the empirical merits of the proposed framework through simulation (Section 5) and through the TripAdvisor prediction task (Section 6) described above. In simulation, we examine our method's robustness to misspecified side information. Quantitative and qualitative comparisons in the TripAdvisor example highlight the advantages of aggregating rare features.

### 1.1. Notation

Given a design matrix $\boldsymbol{X} \in \mathbb{R}^{n\times p}$, let $\boldsymbol{x}_i \in \mathbb{R}^p$ denote the feature vector of observation $i$ and $\boldsymbol{X}_j \in \mathbb{R}^n$ denote the $j$th feature, where $i = 1,\dots,n$ and $j = 1,\dots,p$. For a vector $\boldsymbol{\beta} \in \mathbb{R}^p$, let $\mathrm{supp}(\boldsymbol{\beta}) \subseteq \{1,\dots,p\}$ denote its support (i.e., the set of indices of nonzero elements). Let $\mathbb{S}_{\pm}(\boldsymbol{\beta}) := \left(\mathrm{sign}(\beta_\ell)\right)_{\ell=1,\dots,p}$ encode the signed support of the vector $\boldsymbol{\beta}$. Let $\mathcal{T}$ be a $p$-leafed tree with root $r$, set of leaves $\mathcal{L}(\mathcal{T}) = \{1,\dots,p\}$, and set of nodes $\mathcal{V}(\mathcal{T})$ of size $|\mathcal{T}|$. Let $\mathcal{T}_u$ be the subtree of $\mathcal{T}$ rooted by $u$ for $u \in \mathcal{V}(\mathcal{T})$. We follow the commonly used notions of *child*, *parent*, *sibling*, *descendant*, and *ancestor* to describe the relationships between nodes of a tree. For a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and subset $B$ of $\{1,\dots,n\}$, let $\boldsymbol{A}_B \in \mathbb{R}^{m\times|B|}$ be the submatrix formed by removing the columns of $\boldsymbol{A}$ not in $B$. Let $S(\beta_\ell,\lambda) = \mathrm{sign}(\beta_\ell)\cdot\max\{|\beta_\ell| - \lambda, 0\}$ be the soft-thresholding operator applied to $\beta_\ell \in \mathbb{R}$. We let $\boldsymbol{e}_j$ denote the vector having a one in the $j$th entry and zero elsewhere. For sequences $a_n$ and $b_n$, we write $a_n \lesssim b_n$ to mean that $a_n/b_n$ is eventually bounded, and we write $a_n = o(b_n)$ to mean that $a_n/b_n$ converges to zero.

## 2. Rare Features and the Promise of Aggregation

### 2.1. The Difficulty Posed by Rare Features

Consider the linear model,

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta}^* + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim N(\boldsymbol{0}, \sigma^2\boldsymbol{I}_n), \tag{1}$$

where $\boldsymbol{y} = (y_1,\dots,y_n)^T \in \mathbb{R}^n$ is a response vector, $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ is a design matrix, $\boldsymbol{\beta}^*$ is a $p$-vector of parameters, and $\boldsymbol{\varepsilon} \in \mathbb{R}^n$ is a vector of independent Gaussian errors having variance $\sigma^2$. In this article, we focus on counts data, that is, $\boldsymbol{X}_{ij}$ records the frequency of an event $j$ in observation $i$. In particular, we will assume throughout that $\boldsymbol{X}$ has nonnegative elements.

The lasso (Tibshirani 1996) is an estimator that performs variable selection, making it well-suited to the $p \gg n$ setting:

$$\hat{\boldsymbol{\beta}}_\lambda^{\text{lasso}} \in \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min}\ \frac{1}{2n}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1. \tag{2}$$

When $\lambda = 0$, this coincides with the OLS estimator, which is uniquely defined when $n > p$ and $\boldsymbol{X}$ is full rank:

$$\hat{\boldsymbol{\beta}}^{\text{OLS}}(n) = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}.$$

To better understand the challenge posed by rare features, we begin by considering the effect of a single rare feature on OLS in the classical $p$-fixed, $n \to \infty$ regime. We take the $j$th feature to be a binary vector having $k$ nonzeros, where $k$ is a fixed value not depending on $n$. As $n$ increases, the proportion of nonzero elements, $k/n$, goes to 0. We show in Theorem 1 that $\hat{\beta}_j^{\text{OLS}}(n)$ does not converge in probability to $\beta_j^*$ with increasing sample size. This establishes that OLS is not a consistent estimator of $\boldsymbol{\beta}^*$ even in a $p$-fixed asymptotic regime.

*Theorem 1.* Consider the linear model (1) with $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ having full column rank. Further suppose that $\boldsymbol{X}_j$ is a binary vector having (a constant) $k$ nonzeros. It follows that there exists $\eta > 0$ for which

$$\liminf_{n\to\infty} \mathbb{P}\left(\left|\hat{\beta}_j^{\text{OLS}}(n) - \beta_j^*\right| > \eta\right) > 0.$$

*Proof.* The result follows from taking $\liminf_{n\to\infty}$ of both sides of (1) in Appendix A of the supplementary materials and observing that $2\Phi\left(-\eta k^{1/2}/\sigma\right)$ does not depend on $n$. □

The above result highlights the difficulty of estimating the coefficient of a rare feature. This suggests that even when rare features are not explicitly discarded, variable selection methods may fail to ever select them regardless of their strength of association with the response. Other researchers have also acknowledged the difficulty posed by rare features in different scenarios. For example, in the context of hypothesis testing for high-dimensional sparse binary regression, Mukherjee, Pillai, and Lin (2015) showed that when the design matrix is too sparse, any test has no power asymptotically, and signals cannot be detected regardless of their strength. Since the failure is caused by the sparsity of the features, it is therefore natural to ask if "densifying the features" in an appropriate way would fix the problem. As discussed above, aggregating the counts of related events may be a reasonable way to allow a method to make use of the information in rare features.

### 2.2. Aggregating Rare Features Can Help

Given $m$ subsets of $\{1,\dots,p\}$, we can form $m$ aggregated features by summing within each subset. We can encode these subsets in a binary matrix $\boldsymbol{A} \in \{0,1\}^{p\times m}$ and form a new design matrix of aggregated features as $\widetilde{\boldsymbol{X}} = \boldsymbol{X}\boldsymbol{A}$. The columns of $\widetilde{\boldsymbol{X}}$ are also counts, but represent the frequency of $m$ different *unions* of the $p$ original events. For example, if the first subset is $\{1,6,8\}$, the first column of $\boldsymbol{A}$ would be $\boldsymbol{e}_1 + \boldsymbol{e}_6 + \boldsymbol{e}_8$ and the first aggregated feature would be $\widetilde{\boldsymbol{X}}_1 = \boldsymbol{X}_1 + \boldsymbol{X}_6 + \boldsymbol{X}_8$, recording the number of times any of the first, sixth, or eighth events occur. A linear model, $\widetilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}}$, based on the aggregated features can be equivalently expressed as a linear model, $\boldsymbol{X}\boldsymbol{\beta}$, in terms of the original features as long as $\boldsymbol{\beta}$ satisfies a set of linear constraints (ensuring that it is in the column space of $\boldsymbol{A}$):

$$\widetilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}} = (\boldsymbol{X}\boldsymbol{A})\tilde{\boldsymbol{\beta}} = \boldsymbol{X}(\boldsymbol{A}\tilde{\boldsymbol{\beta}}) = \boldsymbol{X}\boldsymbol{\beta}.$$

The vector $\boldsymbol{\beta}$ lies in the column space of $\boldsymbol{A}$ precisely when it is constant within each of the $m$ subsets. For example,

$$\text{enforcing } \beta_1 = \beta_6 = \beta_8 \quad \Leftrightarrow \quad \text{aggregating features:}$$
$$\boldsymbol{X}_1\beta_1 + \boldsymbol{X}_6\beta_6 + \boldsymbol{X}_8\beta_8 = (\boldsymbol{X}_1 + \boldsymbol{X}_6 + \boldsymbol{X}_8)\beta_1 = \widetilde{\boldsymbol{X}}_1\tilde{\beta}_1. \tag{3}$$

In practice, determining how to aggregate features is a challenging problem, and our proposed strategy in Section 3 will use side information to guide this aggregation.

For now, to understand the potential gains achievable by aggregation, we consider an idealized case in which the correct aggregation of features is given to us by an oracle. In the next theorem, we construct a situation in which (a) the lasso on the original rare features is unable to correctly recover the support of $\boldsymbol{\beta}^*$ for any value of the tuning parameter $\lambda$, and (b) an oracle-aggregation of features makes it possible for the lasso to recover the support of $\boldsymbol{\beta}^*$. For simplicity, we take $\boldsymbol{X}$ as a binary matrix, which corresponds to the case in which every feature has $n/p$ nonzero observations. We take $\boldsymbol{\beta}^*$ to have $k$ blocks of size $p/k$, with entries that are constant within each block. The last block is all zeros and the minimal nonzero $|\beta_j^*|$ is restricted to lie within a range that is expands with $n$, $p$, and $k$. The oracle approach delivers to the lasso the $k$ aggregated features that match the structure in $\boldsymbol{\beta}^*$. These aggregated features have $n/k$ nonzeros, and thus are not rare features. Having performed the lasso on these aggregated features, we then duplicate the $k$ elements, $p/k$ times per group, to get $\hat{\boldsymbol{\beta}}_\lambda^{\text{oracle}} \in \mathbb{R}^p$. The lasso with the oracle-aggregator is shown to achieve high-probability signed support recovery whereas the lasso on the original features fails to achieve this property for all values of the tuning parameter $\lambda$.

*Theorem 2.* Consider the linear model (1) with binary matrix $\boldsymbol{X} \in \{0,1\}^{n\times p}$, $p \le n$, and $\boldsymbol{X}^T\boldsymbol{X} = (n/p)\boldsymbol{I}_p$: every column of $\boldsymbol{X}$ has $n/p$ one's and $\boldsymbol{X}\mathbf{1}_p = \mathbf{1}_n$. Suppose $\boldsymbol{\beta}^* = \tilde{\boldsymbol{\beta}}^* \otimes \mathbf{1}_{p/k}$ for $\tilde{\boldsymbol{\beta}}^* = (\tilde{\beta}_1^*, \dots, \tilde{\beta}_{k-1}^*, 0)$. Suppose $k < p/(36\log n)$. Then, the interval $\mathcal{I} = (\sigma\sqrt{\frac{4k}{n}\log(k^2 n)},\ \sigma\sqrt{\frac{p}{3n}\log\left(2\tilde{c}p/k\right)}]$ with $\tilde{c} = \frac{1}{3}e^{(\pi/2+2)^{-1}}\sqrt{\frac{1}{4}+\frac{1}{\pi}}$ is nonempty and for $\min_{i=1,\dots,k-1}\left|\tilde{\beta}_i^*\right| \in \mathcal{I}$, the following two statements hold:

1. The lasso fails to get high-probability signed support recovery:
$$\limsup_{p\to\infty}\ \sup_{\lambda\ge 0} \mathbb{P}\left(\mathbb{S}_\pm(\hat{\boldsymbol{\beta}}_\lambda^{\text{lasso}}) = \mathbb{S}_\pm(\boldsymbol{\beta}^*)\right) \le \frac{1}{e}.$$
2. The lasso with an oracle-aggregation of features succeeds in recovering the correct signed support for some $\lambda > 0$:
$$\lim_{n\to\infty} \mathbb{P}\left(\mathbb{S}_\pm(\hat{\boldsymbol{\beta}}_\lambda^{\text{oracle}}) = \mathbb{S}_\pm(\boldsymbol{\beta}^*)\right) = 1.$$

*Proof.* See Appendix B in the supplementary materials. □

Even when the true model does not have a small number of aggregated features (i.e., $\boldsymbol{\beta}^*$ does not have $k \ll p$ distinct values), it may still be beneficial to aggregate. The next result exhibits a bias-variance tradeoff for feature aggregation.

*Theorem 3 (Bias-variance tradeoff for feature aggregated least squares).* Consider the linear model (1) with $\boldsymbol{X}$ having full column rank ($n > p$) and a general vector $\boldsymbol{\beta}^* \in \mathbb{R}^p$. Let $\mathcal{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_{|\mathcal{C}|}\}$ be an *arbitrary* partition of $\{1, \dots, p\}$. Let $\hat{\boldsymbol{\beta}}^{\mathcal{C}} \in \mathbb{R}^p$ be the estimator formed by performing least squares subject to the constraint that parameters are constant within the groups defined by $\mathcal{C}$. Then, the following mean squared estimation result holds

$$\frac{1}{n}\mathbb{E}\|\boldsymbol{X}\hat{\boldsymbol{\beta}}^{\mathcal{C}} - \boldsymbol{X}\boldsymbol{\beta}^*\|^2 = \frac{1}{n}\|\boldsymbol{X}\|_{op}^2 \sum_{\ell=1}^{|\mathcal{C}|}\sum_{j\in\mathcal{C}_\ell} \left(\beta_j^* - |\mathcal{C}_\ell|^{-1}\sum_{j'\in\mathcal{C}_\ell}\beta_{j'}^*\right)^2 + \frac{\sigma^2|\mathcal{C}|}{n}$$

*Proof.* See Appendix C in the supplementary materials. □

The bias term is small when there is a small amount of variability in coefficient values within groups of the partition $\mathcal{C}$, that is, when $\boldsymbol{\beta}^*$ is *approximately* constant within each group. Even when $\boldsymbol{\beta}^*$ in truth has a large $k$ (even $k = p$), there may still exist a partition $\mathcal{C}$ with $|\mathcal{C}| \ll k$ for which the bias term is small and thus $\mathbb{E}\|\boldsymbol{X}\hat{\boldsymbol{\beta}}^{\mathcal{C}} - \boldsymbol{X}\boldsymbol{\beta}^*\|^2 \ll \mathbb{E}\|\boldsymbol{X}\hat{\boldsymbol{\beta}}^{\text{OLS}} - \boldsymbol{X}\boldsymbol{\beta}^*\|^2 = \sigma^2 p$.

## 3. Main Proposal: Tree-Guided Aggregation

In the previous section, we have seen the potential gains achievable through aggregating rare features. In this section, we propose a tree-guided method for aggregating and selecting rare features. We discuss this tree in Section 3.1, introduce a tree-based parameterization strategy in Section 3.2, and propose a new estimator in Section 3.3.

### 3.1. A Tree to Guide Aggregation

To form aggregated variables, it is infeasible to consider all possible partitions of the features $\{1, \dots, p\}$. Rather, we will consider a tree $\mathcal{T}$ with leaves $1, \dots, p$ and restrict ourselves to partitions that can be expressed as a collection of branches of $\mathcal{T}$ (see, e.g., Figure 1). We sum features within a branch to form our new aggregated features.

We would like to aggregate features that are related, and thus we would like to have $\mathcal{T}$ encode feature similarity information. Such information about the features comes from prior knowledge and/or data sources external to the current regression problem (i.e., not from $y$ and $X$). For example, for microbiome data, $\mathcal{T}$ could be the phylogenetic tree encoding evolutionary relationships among the OTUs (e.g., Matsen, Kodner, and Armbrust 2010; Tang, Li, and Niclolae 2016; Wang and Zhao 2017a) or the co-occurrence of OTUs from past datasets. When features correspond to words, closeness in meaning can be used to form $\mathcal{T}$ (e.g., in Section 6, we perform hierarchical clustering on word embeddings that were learned from an enormous corpus).

In (3), we demonstrated how aggregating a set of features is equivalent to setting these features' coefficients to be equal. To perform tree-guided aggregation, we therefore associate a coefficient $\beta_j$ with each leaf of $\mathcal{T}$ and "fuse" (i.e., set equal to each other) any coefficients within a branch that we wish to aggregate.

### 3.2. A Tree-Based Parameterization

To fuse $\beta_j$'s within a branch, we adopt a tree-based parameterization by assigning a parameter $\gamma_u$ to each node $u$ in $\mathcal{T}$ (this includes both leaves and interior nodes). The left panel of Figure 2 gives an example. Let ancestor$(j) \cup \{j\}$ be the set of nodes in the path from the root of $\mathcal{T}$ to the $j$th feature, which is associated with the $j$th leaf. We express $\beta_j$ as the sum of all the $\gamma_u$'s on the path:

$$\beta_j = \sum_{u\in\text{ancestor}(j)\cup\{j\}} \gamma_u. \tag{4}$$

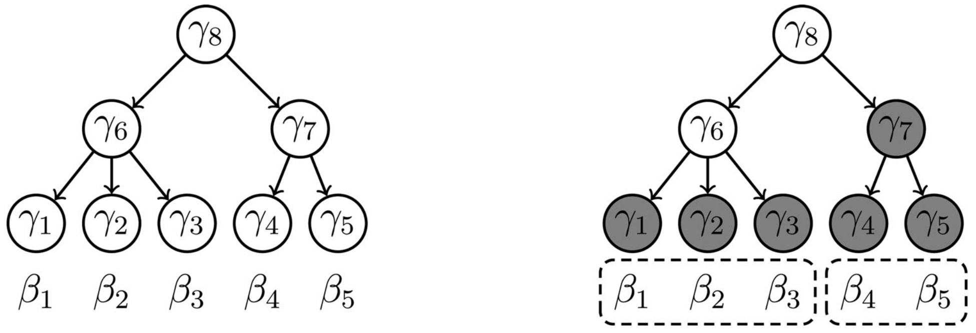

**Figure 2.** (Left) An example of $\boldsymbol{\beta} \in \mathbb{R}^5$ and $\mathcal{T}$ that relates the corresponding five features. By (4), we have $\beta_i = \gamma_i + \gamma_6 + \gamma_8$ for $i = 1, 2, 3$ and $\beta_j = \gamma_j + \gamma_7 + \gamma_8$ for $j = 4, 5$. (Right) By zeroing out the $\gamma_i$'s in the gray nodes, we aggregate $\beta$ into two groups indicated by the dashed contours: $\beta_1 = \beta_2 = \beta_3 = \gamma_6 + \gamma_8$ and $\beta_4 = \beta_5 = \gamma_8$. Counts data are aggregated for features sharing the same coefficient: $\boldsymbol{X\beta} = (\boldsymbol{X}_1 + \boldsymbol{X}_2 + \boldsymbol{X}_3)\beta_1 + (\boldsymbol{X}_4 + \boldsymbol{X}_5)\beta_4$.

This can be written more compactly as $\boldsymbol{\beta} = \boldsymbol{A\gamma}$, where $\boldsymbol{A} \in \{0,1\}^{p\times|\mathcal{T}|}$ is a binary matrix with $A_{jk} := 1_{\{u_k \in \text{ancestor}(j)\cup\{j\}\}} = 1_{\{j\in\text{descendant}(u_k)\cup\{u_k\}\}}$. The descendants of each node $u$ define a branch $\mathcal{T}_u$, and zeroing out $\gamma_v$'s for all $v \in \text{descendant}(u)$ fuses the coefficients in this branch, that is, $\{\beta_j : j \in \mathcal{L}(\mathcal{T}_u)\}$. Thus, $\gamma_{\text{descendant}(u)} = 0$ is equivalent to aggregating the features $\boldsymbol{X}_j$ with $j \in \mathcal{L}(\mathcal{T}_u)$ (see the right panel of Figure 2).

Another way of viewing this parameterization's merging of branches is by expressing $\boldsymbol{X\beta} = \boldsymbol{XA\gamma}$, where $(XA)_{ik} = \sum_{j=1}^p X_{ij}A_{jk} = \sum_{j:j\in\text{descendant}(u_k)\cup\{u_k\}} X_{ij}$ aggregates counts over all the descendant features of node $u_k$. By aggregating nearby features, we allow rare features to borrow strength from their neighbors, allowing us to estimate shared coefficient values that would otherwise be too difficult to estimate. In the next section, we describe an optimization problem that uses the $\boldsymbol{\gamma}$ parameterization to simultaneously perform feature aggregation and selection.

### 3.3. The Optimization Problem

Our proposed estimator $\hat{\boldsymbol{\beta}}$ is the solution to the following convex optimization problem:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p,\boldsymbol{\gamma}\in\mathbb{R}^{|\mathcal{T}|}} \left\{ \frac{1}{2n}\left\|\boldsymbol{y} - \boldsymbol{X\beta}\right\|_2^2 + \lambda\left(\alpha\sum_{\ell=1}^{|\mathcal{T}|} w_\ell\left|\gamma_\ell\right| + (1-\alpha)\sum_{j=1}^p \tilde{w}_j\left|\beta_j\right|\right) \quad \text{s.t. } \boldsymbol{\beta} = \boldsymbol{A\gamma}\right\}. \tag{5}$$

We apply a weighted $\ell_1$ penalty to induce sparsity in $\hat{\boldsymbol{\gamma}}$, which in turn induces fusion of the coefficients in $\hat{\boldsymbol{\beta}}$. In the high-dimensional setting, sparsity in feature coefficients is also desirable, so we also apply a weighted $\ell_1$ penalty on $\boldsymbol{\beta}$. The tuning parameter $\lambda$ controls the overall penalization level while $\alpha$ determines the trade-off between the two types of regularization: fusion and sparsity. In practice, both $\lambda$ and $\alpha$ are determined via cross-validation. The choice of weights is left to the user. Our theoretical results (Section 4) suggest a particular choice of weights, although in our empirical studies (Sections 5 and 6) we simply take all weights to be 1 except for the root, which we leave unpenalized ($w_r = 0$). Choosing $w_r = 0$ allows one to apply strong shrinkage of all coefficients toward a common value other than zero.

When $\alpha = 0$, (5) reduces to a lasso problem in $\boldsymbol{\beta}$; when $\alpha = 1$, (5) reduces to a lasso problem in $\boldsymbol{\gamma}$. Both extreme cases can be efficiently solved with a lasso solver such as `glmnet` (Friedman, Hastie, and Tibshirani 2010). For $\alpha \in (0,1)$, (5) is a generalized lasso problem (Tibshirani and Taylor 2011) in $\boldsymbol{\gamma}$, and can be solved in principle using preexisting solvers (e.g., Arnold and Tibshirani 2014). However, better computational performance, in particular in high-dimensional settings, can be attained using an algorithm specially tailored to our problem. With weights $w_r = 0$ and $\{w_\ell = 1, \tilde{w}_j = 1\}_{\{\ell\neq r, j\in[1,p]\}}$, we write (5) as a *global consensus problem* and solve this using alternating direction method of multipliers (ADMM, Boyd et al. 2011). The consensus problem introduces additional copies of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, which decouples the various parts of the problem, leading to efficient ADMM updates

$$\min_{\substack{\boldsymbol{\beta}^{(1)},\boldsymbol{\beta}^{(2)},\boldsymbol{\beta}^{(3)},\boldsymbol{\beta}\in\mathbb{R}^p\\ \text{and } \boldsymbol{\gamma}^{(1)},\boldsymbol{\gamma}^{(2)},\boldsymbol{\gamma}\in\mathbb{R}^{|\mathcal{T}|}}} \left\{ \frac{1}{2n}\left\|\boldsymbol{y} - \boldsymbol{X\beta}^{(1)}\right\|_2^2 + \lambda\left(\alpha\sum_{\ell=1}^{|\mathcal{T}|} w_\ell\left|\gamma_\ell^{(1)}\right| + (1-\alpha)\sum_{j=1}^p \tilde{w}_j\left|\beta_j^{(2)}\right|\right) \right. \tag{6}$$
$$\left. \text{s.t. } \boldsymbol{\beta}^{(3)} = \boldsymbol{A\gamma}^{(2)}, \boldsymbol{\beta} = \boldsymbol{\beta}^{(1)} = \boldsymbol{\beta}^{(2)} = \boldsymbol{\beta}^{(3)} \text{ and } \boldsymbol{\gamma} = \boldsymbol{\gamma}^{(1)} = \boldsymbol{\gamma}^{(2)} \right\}.$$

In particular, our ADMM approach requires performing a singular value decomposition (SVD) on $\boldsymbol{X}$, an SVD on $(\boldsymbol{I}_p : -\boldsymbol{A})$ (these are reused for all $\lambda$ and $\alpha$), and then applying matrix multiplies and soft-thresholdings until convergence. See Algorithm 1 in Appendix D of the supplementary materials for details. Appendix D.1 in the supplementary materials provides a derivation of Algorithm 1 and Appendix D.2 in the supplementary materials discusses a slight modification for including an intercept, which is desirable in practice.

### 3.4. Connections to Other Work

Before proceeding, we draw some connections to other work. Wang and Zhao (2017b) introduced a penalized regression method with high-dimensional and compositional covariates that uses a phylogenetic tree. They apply an $\ell_1$ penalty on the sum of coefficients within a subtree, for every possible subtree in

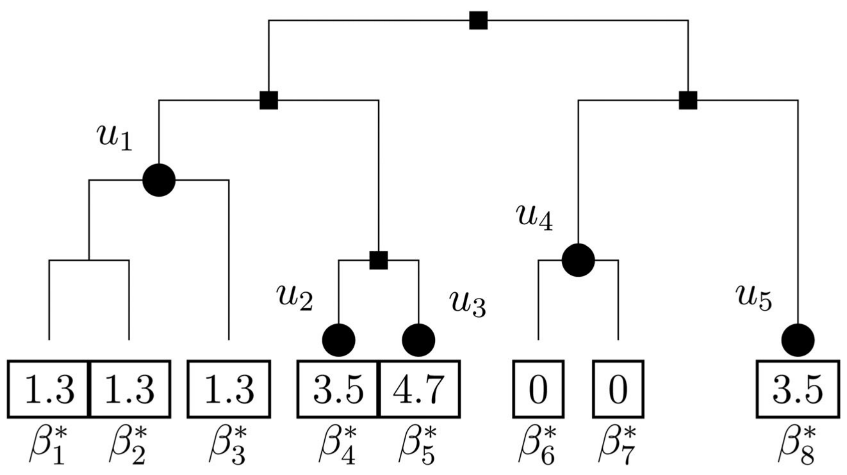

**Figure 3.** In the above tree, $B^* = \{u_1, u_2, u_3, u_4, y_5\}$ has its nodes labeled with black circles.

the phylogenetic tree. Their penalty is designed to encourage the sum of coefficients within a subtree to be zero, which naturally detects a subcomposition of microbiome features. By contrast, our method applies an $\ell_1$ penalty on our latent variables $\gamma_u$'s, which induces the regression coefficients within subtrees to have equal values. Thus, their penalty encourages the sum of coefficients within a subtree to be zero, whereas ours induces equality. For a simple example, suppose $\beta_1$ and $\beta_2$ form a subtree of the phylogenetic tree. Their penalty would promote $\beta_1 = -\beta_2$ whereas ours would promote $\beta_1 = \beta_2$. The basic assumption of their method is that *contrasts* of species abundances within a subtree may be predictive, whereas the basic assumption of our method is that *average* species abundance within a subtree may be predictive. These different structural assumptions will be appropriate in different situations. In this article's context of rare features, our assumption is the relevant one: consider a subtree containing a large number of species. By asking for equality of coefficients, our method promotes the aggregation of species counts across the subtree, leading to denser features; by contrast, asking for coefficients to sum to zero does not address the problem of feature rarity.

Existing work has also considered the setting in which regression coefficients are thought to be clustered into a small number of groups of equal coefficient value. For example, She (2010) and Ke, Fan, and Wu (2015) did this by penalizing coefficient differences. Neither method, however, is focused specifically on rare features and thus they do not rely on side information to perform this clustering of coefficients. In our setting, the side information provided by the tree plays an important role in compensating for the extremely small amount of information available about rare features.

Several other methods assume a relevant undirected graph over the predictors and use a graph-Laplacian or graph-total-variation penalty to promote equality of coefficients that are nearby on the graph (Li and Li 2010; Li, Raskutti, and Willett 2018). Depending on the setting, this graph may either be pure side information (Li and Li 2010) or be a covariance graph estimated based on $X$ itself (Li, Raskutti, and Willett 2018). While the above methods use graph information "edge-by-edge," Yu and Liu (2016) incorporated graphical information "node-by-node" to promote joint selection of predictors that are connected on the graph.

Guinot et al. (2017) considered a similar idea of aggregating genomic features with the help of a hierarchical clustering tree; however, the tree is learned from the design matrix and the prediction task is only used to determine the level of tree cut, whereas our method in effect uses the response to flexibly choose differing aggregation levels across the tree. We consider a strategy similar to theirs, which we call `L1-ag-h` in the empirical comparisons. Kim and Xing (2012) proposed a tree-guided group lasso approach in the context of multi-response regression. In their context, the tree relates the different responses and is used to borrow strength across related prediction tasks. Zhai et al. (2018) proposed a variance component selection scheme that aggregates OTUs to clusters at higher phylogenetic level, and treats the aggregated taxonomic clusters as multiple random effects in a variance component model. Finally, Khabbazian et al. (2016) proposed a phylogenetic lasso method to study trait evolution from comparative data and detect past changes in the expected mean trait values.

## 4. Statistical Theory

In this section, we study the prediction consistency of our method. Since $\mathcal{T}$ encodes feature similarity information, throughout the section we require $\mathcal{T}$ to be a "full" tree such that each node is either a leaf or possesses at least two child nodes. We begin with some definitions.

*Definition 1.* We say that $B \subseteq \mathcal{V}(\mathcal{T})$ is an *aggregating set* with respect to $\mathcal{T}$ if $\{\mathcal{L}(\mathcal{T}_u) : u \in B\}$ forms a partition of $\mathcal{L}(\mathcal{T})$.

The black circles in Figure 3 form an aggregating set since their branches' leaves are a partition of $\{1, \ldots, 8\}$. We would like to refer to "the true aggregating set $B^*$ with respect to $\mathcal{T}$" and, to do so, we must first establish that there exists a unique coarsest aggregating set corresponding to a vector $\boldsymbol{\beta}^*$.

*Lemma 1.* For any $\boldsymbol{\beta}^* \in \mathbb{R}^p$, there exists a unique coarsest aggregating set $B^* := B(\boldsymbol{\beta}^*, \mathcal{T}) \subseteq \mathcal{V}(\mathcal{T})$ (hereafter "the aggregating set") with respect to the tree $\mathcal{T}$ such that (a) $\beta_j^* = \beta_k^*$ for $j, k \in \mathcal{L}(\mathcal{T}_u)\ \forall u \in B^*$, (b) $|\beta_j^* - \beta_k^*| > 0$ for $j \in \mathcal{L}(\mathcal{T}_u)$ and $k \in \mathcal{L}(\mathcal{T}_v)$ for siblings $u, v \in B^*$.

The lemma (proved in Appendix E of the supplementary materials) defines $B^*$ as the aggregating set such that further merging of siblings would mean that $\beta^*$ is not constant within each subset of the partition.

*Definition 2.* Given the triplet $(\mathcal{T}, \boldsymbol{\beta}^*, \boldsymbol{X})$, we define (a) $\widetilde{\boldsymbol{X}} = \boldsymbol{X}\boldsymbol{A}_{B^*} \in \mathbb{R}^{n\times|B^*|}$ to be the *design matrix of aggregated features*, which uses $B^* = B(\boldsymbol{\beta}^*, \mathcal{T})$ as the aggregating set, and (b) $\tilde{\boldsymbol{\beta}}^* \in \mathbb{R}^{|B^*|}$ to be the coefficient vector using these aggregated features: $\boldsymbol{\beta}^* = \boldsymbol{A}_{B^*}\tilde{\boldsymbol{\beta}}^*$.

We are now ready to provide a bound on the prediction error of our estimator, which is proved in Appendix F of the supplementary materials.

*Theorem 4 (Prediction error bound).* If we take $\lambda \geq 4\sigma\sqrt{\log(2p)/n}$, $\tilde{w}_j = \|\boldsymbol{X}_j\|_2/\sqrt{n}$ for $1 \leq j \leq p$ and $w_\ell = \|\boldsymbol{X}\boldsymbol{A}_\ell\|_2/\sqrt{n}$ for $1 \leq \ell \leq |\mathcal{T}|$, then

$$\frac{1}{n}\left\|\boldsymbol{X}\hat{\boldsymbol{\beta}} - \boldsymbol{X}\boldsymbol{\beta}^*\right\|_2^2 \leq 3\lambda\left((1-\alpha)\sum_{j\in\mathcal{A}^*}\tilde{w}_j|\beta_j^*| + \alpha\sum_{\ell\in B^*} w_\ell|\tilde{\beta}_\ell^*|\right)$$

holds with probability at least $1 - 2/p$ for any $\alpha \in [0, 1]$.

This is a slow rate bound for our method. The standard slow rate bound for the lasso is $\sigma\sqrt{\log p/n}\|\boldsymbol{\beta}^*\|_1$. The next corollary establishes that our method, for any choice of $\alpha$, achieves this rate.

*Corollary 1.* Suppose $\|\boldsymbol{X}_j\|_2 \leq \sqrt{n}$ for $1 \leq j \leq p$. Then, taking $\lambda = 4\sigma\sqrt{\log(2p)/n}$ and using the weights in Theorem 4,

$$\frac{1}{n}\|\boldsymbol{X}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*)\|_2^2 \lesssim \sigma\sqrt{\log p/n}\|\boldsymbol{\beta}^*\|_1$$

holds with probability at least $1 - 2/p$ for any $\alpha \in [0, 1]$.

*Proof.* See Appendix G in the supplementary materials. □

The previous corollary establishes that one does not worsen the rate by using our method over the lasso in a generic setting. But is there an advantage to using our method? Our method is designed to do well in circumstances in which $|B^*|$ is small, that is $\boldsymbol{\beta}^*$ is mostly constant with grouping given by the provided tree. The next corollary considers the extreme case in which $\boldsymbol{\beta}^*$ is constant (and nonzero).

*Corollary 2.* Under the conditions of Corollary 1, suppose $\beta_1^* = \cdots = \beta_p^* \neq 0$. Taking $\alpha \geq \left[1 + \|\boldsymbol{X}\boldsymbol{1}_p\|_2/(p\sqrt{n})\right]^{-1}$,

$$\frac{1}{n}\left\|\boldsymbol{X}\hat{\boldsymbol{\beta}} - \boldsymbol{X}\boldsymbol{\beta}^*\right\|_2^2 \lesssim \sigma\sqrt{\frac{\log p}{n}}\left\|\boldsymbol{\beta}^*\right\|_1 \cdot \frac{\left\|\boldsymbol{X}\boldsymbol{1}_p\right\|_2}{p\sqrt{n}}$$

holds with probability at least $1 - 2/p$. Thus, this improves the lasso rate when $\left\|\boldsymbol{X}\boldsymbol{1}_p\right\|_2 = o(p\sqrt{n})$.

*Proof.* See Appendix H in the supplementary materials. □

For certain sparse designs, the above condition holds. For example, when $n = p$ and $\boldsymbol{X} = \sqrt{n}\boldsymbol{I}_n$, $\|\boldsymbol{X}\boldsymbol{1}_p\|_2 = \sqrt{n}\|\boldsymbol{1}_n\|_2 = n$ which is $o(p\sqrt{n}) = o(n^{3/2})$. The next proposition considers a more general sparse $\boldsymbol{X}$ scenario.

*Proposition 1.* Suppose each column of $\boldsymbol{X}$ has exactly $r$ nonzero entries chosen at random and independently of all other columns. Suppose all nonzero entries equal $\sqrt{n/r}$ so that $\|\boldsymbol{X}_j\|_2 = \sqrt{n}$ for every column $1 \leq j \leq p$. If $\frac{8\log(n+1)}{3p} < \frac{r}{n} \to 0$, then $\|\boldsymbol{X}\boldsymbol{1}_p\|_2/(p\sqrt{n}) \to 0$ in probability.

*Proof.* See Appendix I in the supplementary materials. □

Proposition 1 combined with Corollary 2 demonstrates that our method can improve the prediction error rate over the lasso. While this corollary focuses on the rather extreme case in which all coefficients are equal, we expect the result to be generalizable to a wider class of settings. Indeed, the empirical results of the next section suggest that our method can outperform the lasso in many settings.

## 5. Simulation Study

We start by forming a tree $\mathcal{T}$ with $p$ leaves. To do so, we generate $p$ latent points in $\mathbb{R}$ and then apply hierarchical clustering (using `hclust` in R Core Team (2016) with complete linkage). We would like the tree to partition the leaves into $k$ clusters of varying sizes and at differing heights in the tree (which will correspond to the *true aggregation* of the features). To do so, we first generate $k$ cluster means $\mu_1, \ldots, \mu_k \in \mathbb{R}$ with $\mu_i = 1/i$. The first $k/2$ means have $3p/(2k)$ associated latent points each, and the remaining $k/2$ means have $p/(2k)$ associated latent points each. The latent points associated with $\mu_i$ are drawn independently from $N\left(\mu_i, \tau^2 \min_j(\mu_i - \mu_j)^2\right)$, where $\tau = 0.05$.

By design, there are $k$ interior nodes in $\mathcal{T}$ corresponding to these $k$ groups, which we index by $B^*$. We form $\boldsymbol{A}$ corresponding to this tree and generate $\boldsymbol{\beta}^* = \boldsymbol{A}_{B^*}\tilde{\boldsymbol{\beta}}^*$. We zero out $k \cdot s$ elements of $\tilde{\boldsymbol{\beta}}^* \in \mathbb{R}^k$ and draw the magnitudes of the remaining elements independently from a Uniform(1.5, 2.5) distribution. We alternate signs of the nonzero coefficients of $\tilde{\boldsymbol{\beta}}^*$. The design matrix $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ is simulated from a Poisson(0.02) distribution, and the response $\boldsymbol{y} \in \mathbb{R}^n$ is simulated from (1) with $\sigma = \|\boldsymbol{X}\boldsymbol{\beta}^*\|_2/\sqrt{5n}$. For every method under consideration, we average its performance over 100 repetitions in all the following simulations.

We consider both low-dimensional ($n = 500, p = 100, s = 0$) and high-dimensional ($n = 100, p = 200, s \in \{0.2, 0.6\}$) scenarios, in each case taking a sequence of $k$ values up to $p/2$. We apply our method with $\mathcal{T}$ and vary the tuning parameters $(\alpha, \lambda)$ along an 8-by-50 grid of values. We take all weights equal to 1 except that of the root node, which we take as zero (leaving it unpenalized). In the low-dimensional case, we compare our method to *oracle least squares*, in which we perform least squares on $\boldsymbol{X}\boldsymbol{A}_{B^*}$. Oracle least squares represents the best possible performance of any method that attempts to aggregate features. We also include least squares on the original design matrix $\boldsymbol{X}$. In the high-dimensional case, we compare our method to the *oracle lasso*, in which the true aggregation $\boldsymbol{X}\boldsymbol{A}_{B^*}$ (but not the sparsity in $\tilde{\boldsymbol{\beta}}^*$) is known, and to the lasso and ridge regression, which are each computed across a grid of 50 values of the tuning parameter.

In addition to the above methods, we compare our method to three other approaches, meant to represent variations of how

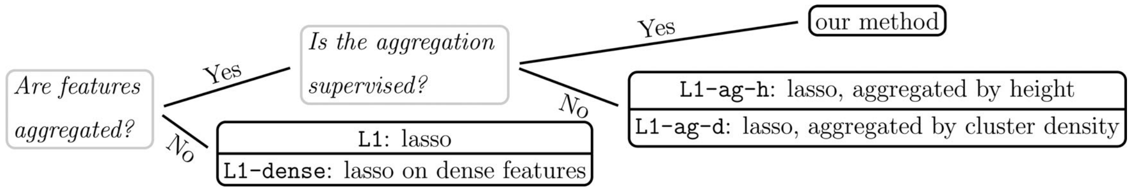
Are features aggregated?
Yes
No
Is the aggregation supervised?
Yes
No
our method
L1-ag-h: lasso, aggregated by height
L1-ag-d: lasso, aggregated by cluster density
L1: lasso
L1-dense: lasso on dense features

**Figure 4.** A comparison between our method and four other methods.

the lasso is typically applied when rare features are present (see Figure 4 for a schematic). The first approach, which we refer to as `L1-dense`, applies the lasso after first discarding any features that are in fewer than 1% of observations. The second and third approaches apply the lasso with features aggregated according to $\mathcal{T}$ in an unsupervised manner. The second approach, `L1-ag-h`, aggregates features that are in the same cluster after cutting the tree at a certain height. In addition to the lasso tuning parameter, the height at which we cut the tree is a second tuning parameter (chosen along an equally spaced grid of eight values). The third approach, `L1-ag-d`, performs merges in a bottom-up fashion along the tree until all aggregated features have density above some threshold. This threshold is an additional tuning parameter (chosen along an equally spaced grid of eight values between 0.01 and 1). The lasso tuning parameter in these methods is always chosen along a grid of 50 values.

We measure the *best mean-squared estimation error*, that is, $\min_\Lambda \|\hat{\boldsymbol{\beta}}(\Lambda) - \boldsymbol{\beta}^*\|_2^2/p$, where "best" is with respect to each method's tuning parameter(s) $\Lambda$. The top two panels of Figure 5 show the performance of the methods in the low-dimensional and high-dimensional scenarios, respectively. Given that our method includes least squares and the lasso as special cases, it is no surprise that our methods have better attainable performance than those methods. These results indicate that our method performs nearly as well as the oracle when the true number of aggregated features, $k$, is small and degrades (along with the oracle) as this quantity increases. The two other methods that use the tree, `L1-ag-h` and `L1-ag-d`, do less well than our method, but still do better than `L1-dense`, which simply discards rare features. In the $(n, p, s) = (100, 200, 0.2)$ case, `L1-dense` performs almost identically to the lasso, while `L1-ag-h` and `L1-ag-d` degrade to the lasso as $k$ increases. By comparing the right two panels of Figure 5, we notice our method outperforms the ridge at large $k$ when $s$ increases from 0.2 to 0.6, which can be explained by the increased sparsity in $\boldsymbol{\beta}^*$.

We also evaluate model performance with respect to *group recovery* and *support recovery*. Recall that our method is computed over eight $\alpha$ values, between 0 and 1, and fifty $\lambda$ values. At each $\alpha$, we find the minimizer $\hat{\boldsymbol{\beta}}(\hat{\lambda})$ for $\|\hat{\boldsymbol{\beta}}(\lambda) - \boldsymbol{\beta}^*\|_2^2$ over all $\lambda$ values. We measure group recovery by computing the Rand index (comparing the grouping of $\hat{\boldsymbol{\beta}}$ to that of $\boldsymbol{\beta}^*$), and measure support recovery by computing the Hamming distance (between the supports of $\hat{\boldsymbol{\beta}}$ to that of $\boldsymbol{\beta}^*$). The closer the Rand index is to one, the better our method recovers the correct groups. The smaller the Hamming distance is, the better our method recovers the correct support. For the high-dimensional scenario with $k = 40$, we plot eight pairs (one for each $\alpha$ value) of Rand index and Hamming distance values for $s = 0.2$ and $s = 0.6$ in Figure 6. We also compute the two metrics for the lasso, ridge, `L1-dense`, `L1-ag-h`, `L1-ag-d`, and oracle lasso. In the left panel of Figure 6, which corresponds to the low-sparsity case in $\boldsymbol{\beta}^*$, our method achieves its best performance at the largest $\alpha$ value. As the sparsity level increases, we see from the right panel of Figure 6 that the best $\alpha$ shifts toward zero. In both cases, our method outperforms the lasso, ridge, `L1-dense`, `L1-ag-h`, and `L1-ag-d`.

Clearly, the performance of our method, `L1-ag-h` and, `L1-ag-d` will depend on the quality of the tree being used. In the previous simulations, we provided our method with a tree that is perfectly compatible with the true aggregating set. In practice, the tree used may be only an approximate representation of how features should be aggregated. We therefore study the sensitivity of our method to misspecification of the tree. We return to the high-dimensional setting above with $k = 40$, and we generate a sequence of trees that are increasingly distorted representations of how the data should in fact be aggregated.

We begin with a true aggregation of the features into $k$ groups as described before. In each repetition of the simulation, we generate a (random) tree $\mathcal{T}$ by performing hierarchical clustering on $p$ random variables generated similarly as before except having increasing $\tau$ value, as a way to control the degradation level of the tree. When $\tau$ is small, the latent variables will be well-separated by group so that the tree will have an aggregating set that matches the true aggregation structure (with high probability). As $\tau \in \{0.05, 0.15, 0.25, 0.35, 0.45\}$ increases, the between-group variability becomes relatively smaller compared to within-group variability, and thus the information provided by the tree becomes increasingly weak. The bottom left panel of Figure 5 shows the degradation of our method as $\tau$ increases when $(n, p, s, k) = (100, 200, 0.2, 40)$. Our method, `L1-ag-h`, and `L1-ag-d` all suffer from a poor-quality tree; the latter two degrade more quickly than ours.

## 6. Application to Hotel Reviews

Wang, Lu, and Zhai (2010) crawled TripAdvisor.com to form a dataset[2] of 235,793 reviews and ratings of 1850 hotels by users between February 14, 2009 and March 15, 2009. While there are several kinds of ratings, we focus on a user's overall rating of the hotel (on a 1–5 scale), which we take as our response. We form a document-term matrix $\boldsymbol{X}$ in which $\boldsymbol{X}_{ij}$ is the number of times the $i$th review uses the $j$th adjective.

We begin by converting words to lower case and keeping only adjectives (as determined by WordNet, Fellbaum 1998; Wallace 2007; Feinerer and Hornik 2016). After removing reviews with missing ratings, we are left with 209,987 reviews and 7787

[2]Data source: *http://times.cs.uiuc.edu/~wang296/Data/*

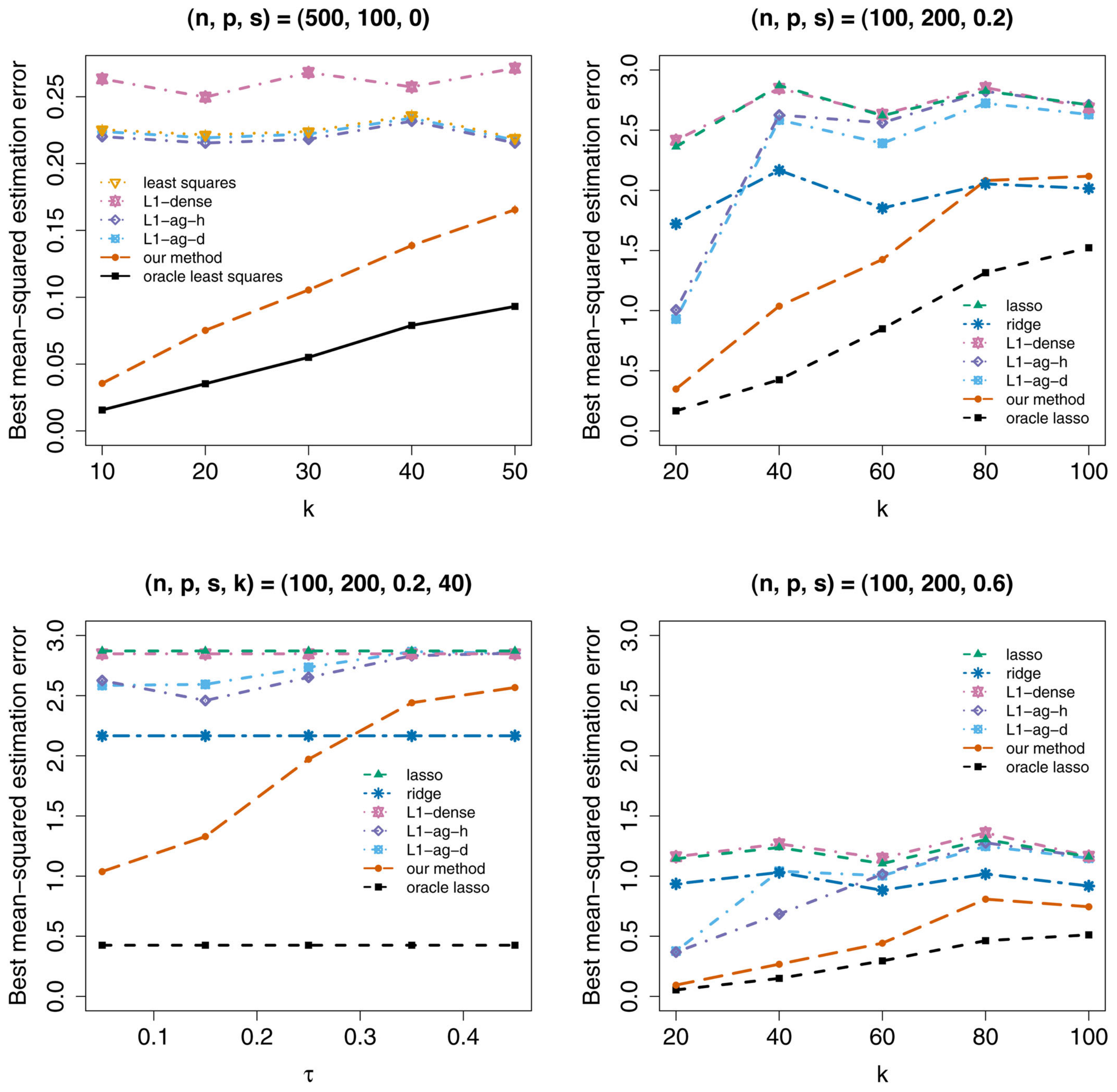

**Figure 5.** Estimation error of all methods under (top left) $(n, p, s) = (500, 100, 0)$ versus varying $k \in \{10, 20, 30, 40, 50\}$; (top right) $(n, p, s) = (100, 200, 0.2)$ versus varying $k \in \{20, 40, 60, 80, 100\}$; (bottom left) $(n, p, s, k) = (100, 200, 0.2, 40)$ versus varying $\tau \in \{0.05, 0.15, 0.25, 0.35, 0.45\}$; (bottom right) $(n, p, s) = (100, 200, 0.6)$ versus varying $k \in \{20, 40, 60, 80, 100\}$.

distinct adjectives. The left panel of Figure 7 shows the distribution of ratings in the data: nearly three quarters of all ratings are above 3 stars. The extremely right-skewed distribution in the right panel of Figure 7 shows that all but a small number of adjectives are highly rare (e.g., over 90% of adjectives are used in fewer than 0.5% of reviews).

Rather than discard this large number of rare adjectives, our method aims to make productive use of these by leveraging side information about the relationship between adjectives. We construct a tree capturing adjective similarity as follows. We start with word embeddings[3] in a 100-dimensional space that were pretrained by *GloVe* (Pennington, Socher, and Manning 2014) on the Gigaword5 and Wikipedia2014 corpora. We also obtain a list of adjectives, which the NRC Emotion Lexicon labels as having either positive or negative sentiments (Mohammad and Turney 2013). We use five nearest neighbors classification within the 100-dimensional space of word embeddings to assign labels to the 5795 adjectives that have not been labeled in the NRC Emotion Lexicon. This sentiment separation determines the two main branches of the tree $\mathcal{T}$. Within each branch, we perform hierarchical clustering of the word embedding vectors. Figure 8 depicts such a tree with 2397 adjectives (as leaves).

We compare our method (with weights all equal to 1 except for the root node, which is left unpenalized) to four other

[3]Data source: *http://nlp.stanford.edu/data/glove.6B.zip*

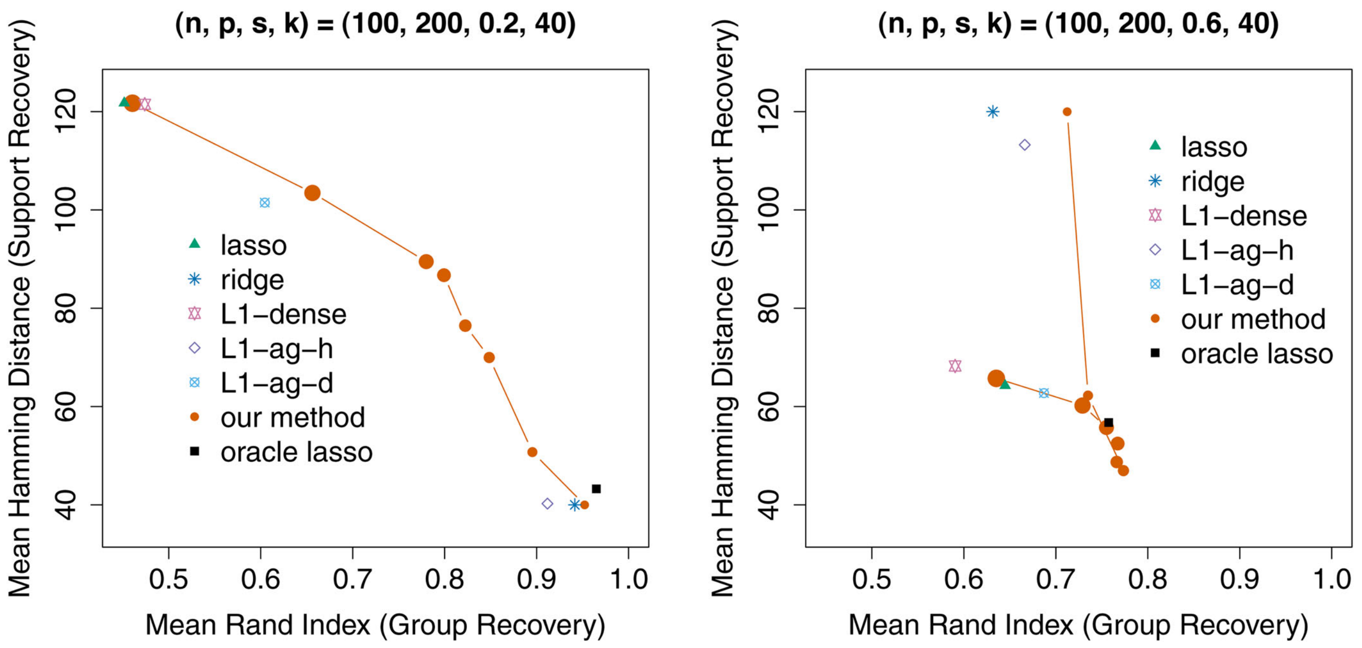

Figure 6. Mean Hamming distance versus mean Rand index for various methods' best estimates under (left) $(n, p, s, k) = (100, 200, 0.2, 40)$ and (right) $(n, p, s, k) = (100, 200, 0.6, 40)$. For our method, the eight circles correspond to eight different $\alpha$ values, varying from 0 to 1. Decreasing circle size corresponds to increasing $\alpha$ value.

| Rating | Proportion |
|---|---|
| 1 | 0.066 |
| 2 | 0.085 |
| 3 | 0.105 |
| 4 | 0.308 |
| 5 | 0.436 |

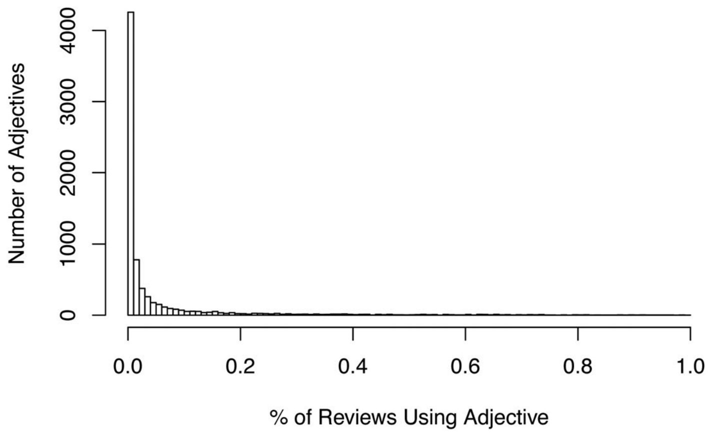

Figure 7. (Left) Distribution of TripAdvisor ratings. (Right) Only 414 adjectives appear in more than 1% of reviews; the histogram gives the distribution of usage-percentages for those adjectives appearing in fewer than 1% of reviews.

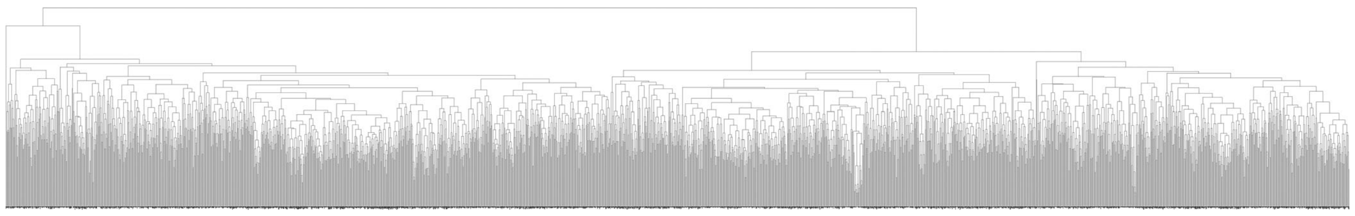

Figure 8. Tree $\mathcal{T}$ over 2397 adjectives: the left subtree is for adjectives with negative sentiment and the right subtree is for adjectives with positive sentiment.

approaches described in Figure 4. For L1-dense, we first discard any adjectives that are in fewer than 0.5% of reviews before applying the lasso. Both L1-ag-h and L1-ag-d have an additional tuning parameter to take care of: for L1-ag-h we vary the height at which we cut the tree along an equally spaced grid of ten values; for L1-ag-d we choose the threshold for aggregations' density along an equally spaced grid of ten values between 0.001 and 0.1.

We hold out 40,000 ratings and reviews as a test set. To observe the performance of these methods over a range of training set sizes, we consider a nested sequence of training sets, ranging from 1% to 100% of the reviews not included in the test set. For all methods, we use 5-fold cross-validation to select tuning parameters and threshold all predicted ratings to be within the interval [1, 5]. Table 1 displays the mean squared prediction error (MSPE) on the test set for each method and training set size.

As the size of the training set increases, all methods except for the lasso with aggregation based on density (L1-ag-d) achieve lower MSPE. Among the four lasso-related methods,

Table 1. Performance of five methods on the held-out test set: `L1` is the lasso; `L1-dense` is the lasso on only dense features; `L1-ag-h` is the lasso with features aggregated based on height; and `L1-ag-d` is the lasso with features aggregated based on density level.

| | | | | Mean squared prediction error | | | | |
|---|---|---|---|---|---|---|---|---|
| Prop. | $n$ | $p$ | $n/p$ | `Our method` | `L1` | `L1-dense` | `L1-ag-h` | `L1-ag-d` |
| 1% | 1700 | 2397 | 0.71 | **0.870** | 0.894 | 0.895 | 0.882 | 0.971 |
| 5% | 8499 | 3962 | 2.15 | **0.783** | 0.790 | 0.805 | 0.785 | 0.899 |
| 10% | 16,999 | 4786 | 3.55 | **0.758** | 0.764 | 0.788 | 0.764 | 0.902 |
| 20% | 33,997 | 5621 | 6.05 | **0.742** | 0.749 | 0.773 | 0.747 | 1.173 |
| 40% | 67,995 | 6472 | 10.51 | **0.739** | 0.740 | 0.768 | 0.742 | 1.108 |
| 60% | 101,992 | 6962 | 14.65 | **0.733** | 0.736 | 0.769 | 0.734 | 1.155 |
| 80% | 135,990 | 7294 | 18.64 | **0.733** | **0.733** | 0.765 | 0.734 | 0.886 |
| 100% | 169,987 | 7573 | 22.45 | **0.729** | 0.731 | 0.765 | 0.731 | 0.956 |

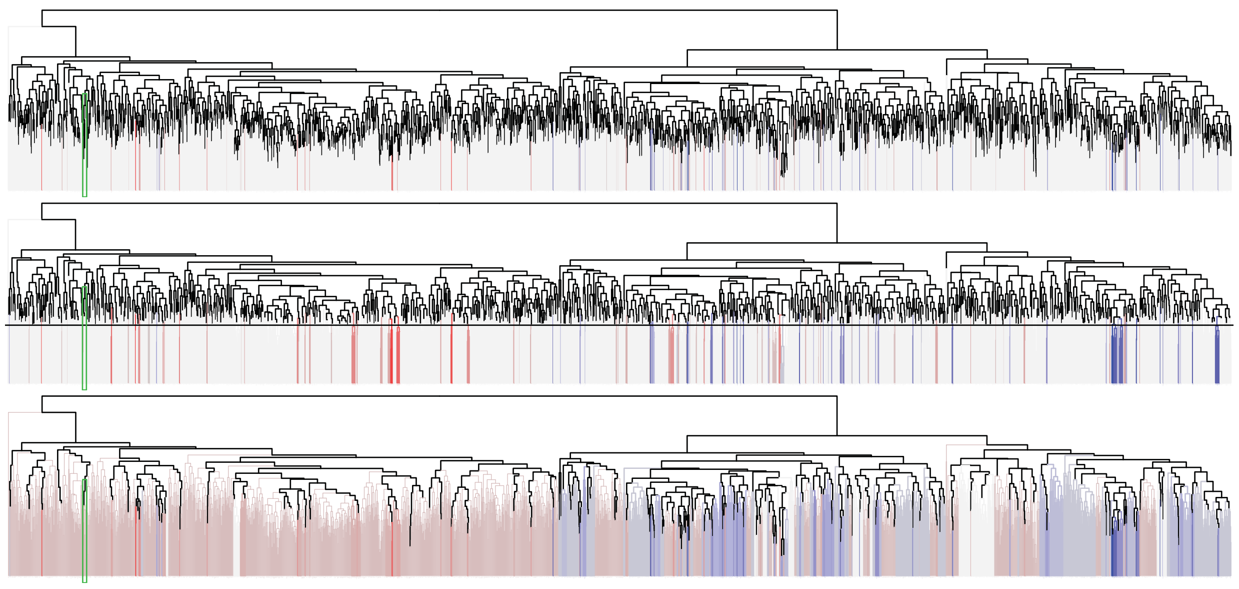

Figure 9. Trees for 2397 adjectives on the leaves with branches colored based on $\hat{\boldsymbol{\beta}}$ estimated with the lasso (top), `L1-ag-h` (middle), and our method (bottom), respectively. Red branch, blue branch, and gray branch correspond to negative, positive, and zero $\hat{\beta}_j$, respectively. Darker color indicates larger magnitude of $\hat{\beta}_j$ and lighter color indicates smaller magnitude of $\hat{\beta}_j$. The horizontal line shown in the middle plot corresponds to the height, chosen from CV, at which `L1-ag-h` cuts the tree and merges the resulting branches.

`L1` and `L1-ag-h` outperform the other two. As the training set size $n$ increases, the number of features $p$ also increase but at a relatively slower rate. We notice that when $n/p$ is less than 10.51, our method outperforms the other four lasso-related methods. As $n/p$ increases beyond 10.51, that is, in the statistically easier regimes, `L1` and `L1-ag-h` attain performance comparable to our method. We conduct paired $t$-tests between squared prediction errors from our method and `L1-ag-h` at every $(n, p)$ pair (i.e., every row of Table 1). Five out of the eight tests are significant at the 0.005 significance level (see Table J.1 in Appendix J of the supplementary materials).

To better understand the difference between our method, the lasso, and `L1-ag-h`, we color the branches of the tree generated in the $n = 1700$ and $p = 2397$ case (i.e., proportion is 1%) according to the sign and magnitude of $\hat{\boldsymbol{\beta}}$ for the three methods. The bottom tree in Figure 9 corresponds to our method and has many nearby branches sharing the same red/blue color, indicating that the corresponding adjective counts have been merged. By contrast, the top tree in Figure 9, which corresponds to the lasso, shows that the solution is sparser and does not have branches of similar color. The middle tree in Figure 9 shows that `L1-ag-h` produces something between the two, merging some adjectives with strong signals while keeping the rest as singletons. The strength and pattern of aggregations vary between our method and `L1-ag-h`. Inspection of the merged branches from our method reveals words of similar meaning and sentiment being aggregated. In Figure 10, we plot $\{|\hat{\beta}_j|\}$ against the percentage of reviews containing an adjective. We find that our method selects rare words more than the other two methods. The rarest word selected by the lasso is "filthy," which appears in 0.47% of reviews. By contrast, our method selects many words that are far more rare: at the extreme of rarest words, our method selects 797 words that appear in only 0.059% of reviews; `L1-ag-h` selects 31 out of these 797 rarest words. Our method is able to select more rare words through aggregation: it aggregates 2244 words into 224 clusters, leaving the remaining 153 words as singletons. Over 70% of these singletons are dense words (where, for this discussion, we call a word "dense" if it appears in at least 1% of reviews and "rare" otherwise). This is four times higher than the percentage of dense words in the original training data. Of the 224 aggregated clusters, 42% are made up entirely of rare words.

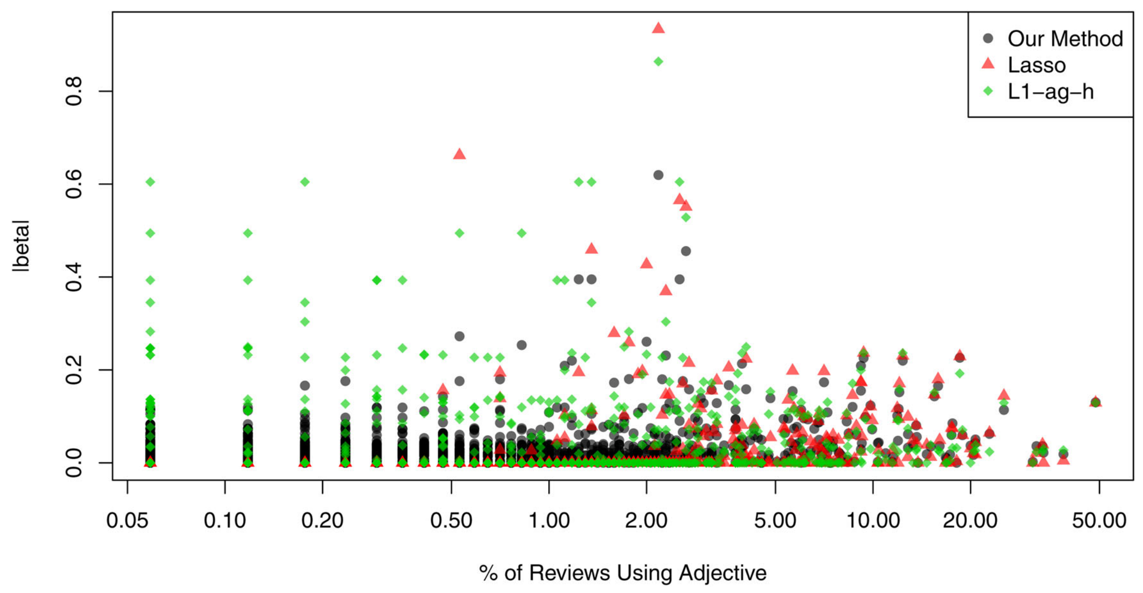

**Figure 10.** Magnitudes of coefficient estimates $\{|\hat{\beta}_j|\}$ versus feature density (on log scale) for our method (black circles), the lasso (red triangles), and `L1-ag-h` (green diamonds).

**Table 2.** Term density and estimated coefficient for adjectives in the selected group.

| Adjectives | Heard | Loud | Yelled | Shouted | Screaming | Crying | Blaring | Banging |
|---|---|---|---|---|---|---|---|---|
| Density[a] | 0.0300 | 0.0235 | 0.0006 | 0.0006 | 0.0029 | 0.0006 | 0.0006 | 0.0041 |
| $\hat{\boldsymbol{\beta}}_{\lambda}^{\text{lasso}}$ | −0.057 | −0.147 | 0 | 0 | 0 | 0 | 0 | 0 |
| $\hat{\boldsymbol{\beta}}_{\lambda,\text{height}}^{\texttt{L1-ag-h}}$ | −0.174 | −0.174 | 0 | 0 | 0 | 0 | 0 | 0 |
| $\hat{\boldsymbol{\beta}}_{\lambda}^{\text{ours}}$ | −0.128 | −0.128 | −0.039 | −0.039 | −0.039 | −0.039 | −0.039 | −0.039 |

[a]The term density is computed over the training set.

After aggregation, over half of these clusters become dense features. As a comparison, `L1-ag-h` aggregates 1339 words into 506 clusters while keeping 1058 words as singletons. Only 10% of these singletons from `L1-ag-h` are dense words. Of the 506 aggregated clusters from `L1-ag-h`, 68% are made by rare words only, among which only 15% become dense clusters after aggregation.

Table 2 shows the density and estimated coefficient values for eight words falling in a particular subtree of $\mathcal{T}$. The words "heard" and "loud" occur far more commonly than the other six words. We see that the lasso only selects these two words whereas our method selects all eight words (assigning them negative coefficient values). In contrast, `L1-ag-h`, while also aggregating the two densest words in this branch into a group, does not select the six rare words. Examining the six rare words, it seems quite reasonable that they should receive negative coefficient values. This suggests to us that the lasso's exclusion of these words has to do with their rareness in the dataset rather than their irrelevance to predicting the hotel rating.

Existing work in sentiment analysis uses side information in other ways to improve predictive performance (Thelwall et al. 2010). For example, Wang et al. (2016), working in an SVM framework, formed a directed graph over words that expresses their "sentiment strength" and then requires that the coefficients corresponding to these words honor the ordering that is implied by the directed graph. For example, if word A is stronger (in expressing a certain sentiment) than word B, which in turn is stronger than word C, then their method enforces $\beta_A \geq \beta_B \geq \beta_C$. Such constraints can in some situations have a regularizing effect that may be of use in rare feature settings: for example, if $\beta_A \approx \beta_C$ and B is a rare word, this constraint would help pin down $\beta_B$'s value. However, if $\beta_A$ and $\beta_C$ are very different from each other, the constraint may offer little help in reducing the variance of the estimate of $\beta_B$. Another difference is that the method uses hard constraints that are not controlled by a tuning parameter, so that even when there is strong evidence in the data that a constraint should be violated, this will not be allowed. By contrast, our method shrinks toward the constant-on-subtrees structure without forcing this to be the case.

Section 5 investigated the effect of using a distorted tree (see the bottom left panel of Figure 5). In the context of words there are multiple choices of trees one could use. Table 3 shows the effect of applying our method to different types of trees. Here, we focus on one situation with 33,997 training reviews. We first consider the effect of changing the dimension of the *GloVe* embedding. One might suppose that as the embedding dimension increases the tree becomes more informative. Indeed, one finds that our method and `L1-ag-h` achieve improved performance as this dimension increases. The last method, `L1-ag-d`, has more variability in its performance as the *GloVe* tree varies, making it the poorest performing method. Both `L1` and `L1-dense` do not make use of the tree and thus they are unaffected by changes to the embedding dimension. But a limitation of the methods is that they all are based on the bag-of-words model, and therefore do not take word context into consideration. For example, the models do not differentiate

Table 3. Performance of five methods under various tree settings on the held-out test set.

| Tree setting | Mean squared prediction error | | | | |
|---|---|---|---|---|---|
| | `Our method` | `L1` | `L1-dense` | `L1-ag-h` | `L1-ag-d` |
| *GloVe*-50d | **0.748** | 0.749 | 0.773 | 0.752 | 0.775 |
| *GloVe*-100d | **0.742** | 0.749 | 0.773 | 0.747 | 1.173 |
| *GloVe*-200d | **0.741** | 0.749 | 0.773 | 0.747 | 1.140 |
| ELMo | **0.669** | 0.676 | 0.732 | 0.685 | 0.783 |

NOTE: Among the tree settings, *GloVe*-50d, *GloVe*-100d and *GloVe*-200d correspond to hierarchical clustering trees generated with *GloVe* embeddings of differing dimensions. We collapse over two million ELMo embedding vectors of adjectives into 6001 clusters using mini-batch K-Means clustering (Sculley 2010), then generate a hierarchical tree upon the 6001 cluster centroids. The performance is based on 33,997 training reviews (i.e., corresponding to the 20% row of Table 1).

between the use of "bad" in "the hotel is bad" versus "the hotel is not that bad." To bring context into consideration, we leverage deep contextualized word representations from ELMo (Peters et al. 2018) to generate an auxiliary tree (see Appendix K in the supplementary materials for details). Unlike traditional word embeddings such as *GloVe* that associate one embedding per word, deep word embeddings can capture the meaning of each word based on the surrounding context. From Table 3, we find test errors improve substantially with the ELMo tree for our method, `L1` and `L1-ag-h`. Among all methods and all tree settings, our method with the ELMo tree performs the best. This suggests our method can better leverage the power of contextualized word embeddings than competing methods.

## 7. Conclusion

In this article, we focus on the challenge posed by highly sparse data matrices, which have become increasingly common in many areas, including biology and text mining. While much work has focused on addressing the challenges of high-dimensional data, relatively little attention has been given to the challenges of sparsity in the data. We show, both theoretically and empirically, that not explicitly accounting for the sparsity in the data hurts one's prediction errors and one's ability to perform feature selection. Our proposed method is able to make productive use of highly sparse features by creating new aggregated features based on side information about the original features. In contrast to simpler tree-based aggregation strategies that are occasionally used as a preprocessing step in biological applications, our method adaptively learns the feature aggregation in a supervised manner. In doing so, our methodology not only overcomes the challenges of data sparsity but also produces features that may be of greater relevance to the particular prediction task of interest.

## Supplementary Materials

**Supplementary materials for "Rare Feature Selection in High Dimensions":** Proofs of theorems, lemmas, and corollaries (and intermediate results); ADMM algorithm for our method.

## Acknowledgments

The authors thank Andy Clark for calling our attention to the challenge of rare features.

## Funding

This work was supported by NSF CAREER grant DMS-1653017.

## References

Arnold, T. B., and Tibshirani, R. J. (2014), "genlasso: Path Algorithm for Generalized Lasso Problems," R Package Version 1.3. [5]

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011), "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends in Machine Learning*, 3, 1–122. [5]

Cao, Y., Zhang, A., and Li, H. (2017), "Microbial Composition Estimation From Sparse Count Data," arXiv no. 1706.02380. [1]

Caporaso, J. G., Kuczynski, J., Stombaugh, J., Bittinger, K., Bushman, F. D., Costello, E. K., Fierer, N., Pena, A. G., Goodrich, J. K., Gordon, J. I., Huttley, G. A., Kelley, S. T., Knights, D., Koenig, J. E., Ley, R. E., Lozupone, C. A., McDonald, D., Muegge, B. D., Pirrung, M., Reeder, J., Sevinsky, J. R., Turnbaugh, P. J., Walters, W. A., Widmann, J., Yatsunenko, T., Zaneveld, J., and Knight, R. (2010), "QIIME Allows Analysis of High-Throughput Community Sequencing Data," *Nature Methods*, 7, 335–336. [1]

Chen, J., Bushman, F. D., Lewis, J. D., Wu, G. D., and Li, H. (2013), "Structure-Constrained Sparse Canonical Correlation Analysis With an Application to Microbiome Data Analysis," *Biostatistics*, 14, 244–258. [1]

Feinerer, I., and Hornik, K. (2016), "wordnet: WordNet Interface," R Package Version 0.1-11. [8]

——— (2017), "tm: Text Mining Package," R Package Version 0.7-1. [1]

Fellbaum, C. (1998), *WordNet: An Electronic Lexical Database*, Cambridge, MA: Bradford Books. [8]

Forman, G. (2003), "An Extensive Empirical Study of Feature Selection Metrics for Text Classification," *Journal of Machine Learning Research*, 3, 1289–1305. [1]

Friedman, J., Hastie, T., and Tibshirani, R. J. (2010), "Regularization Paths for Generalized Linear Models via Coordinate Descent," *Journal of Statistical Software*, 33, 1–22. [5]

Guinot, F., Szafranski, M., Ambroise, C., and Samson, F. (2017), "Learning the Optimal Scale for GWAS Through Hierarchical SNP Aggregation," *BMC Bioinformatics*, 19, 1–14. [6]

Huang, A. (2008), "Similarity Measures for Text Document Clustering," in *Proceedings of the Sixth New Zealand Computer Science Research Student Conference* (NZCSRSC2008), Christchurch, New Zealand, pp. 49–56. [1]

Ke, T., Fan, J., and Wu, Y. (2015), "Homogeneity Pursuit," *Journal of the American Statistical Association*, 110, 175–194. [6]

Khabbazian, M., Kriebel, R., Rohe, K., and Ané, C. (2016), "Fast and Accurate Detection of Evolutionary Shifts in Ornstein-Uhlenbeck Models," *Methods in Ecology and Evolution*, 7, 811–824. [6]

Kim, S., and Xing, E. P. (2012), "Tree-Guided Group Lasso for Multi-Response Regression With Structured Sparsity, With an Application to eQTL Mapping," *The Annals of Applied Statistics*, 6, 1095–1117. [6]

Li, C., and Li, H. (2010), "Variable Selection and Regression Analysis for Graph-Structured Covariates With an Application to Genomics," *The Annals of Applied Statistics*, 4, 1498–1516. [6]

Li, Y., Raskutti, G., and Willett, R. (2018), "Graph-Based Regularization for Regression Problems With Highly-Correlated Designs," arXiv no. 1803.07658. [6]

Lin, W., Shi, P., Feng, R., and Li, H. (2014), "Variable Selection in Regression With Compositional Covariates," *Biometrika*, 101, 785–797. [1]

Liu, X., Yu, S., Janssens, F., Glänzel, W., Moreau, Y., and De Moor, B. (2010), "Weighted Hybrid Clustering by Combining Text Mining and Bibliometrics on a Large-Scale Journal Database," *Journal of the Association for Information Science and Technology*, 61, 1105–1119. [1]

Matsen, F. A., Kodner, R. B., and Armbrust, E. V. (2010), "pplacer: Linear Time Maximum-Likelihood and Bayesian Phylogenetic Placement of Sequences Onto a Fixed Reference Tree," *BMC Bioinformatics*, 11, 538. [4]

McMurdie, P. J., and Holmes, S. (2013), "phyloseq: An R Package for Reproducible Interactive Analysis and Graphics of Microbiome Census Data," *PLOS ONE*, 8, 1–11. [1]

Mohammad, S. M., and Turney, P. D. (2013), "Crowdsourcing a Word–Emotion Association Lexicon," *Computational Intelligence*, 29, 436–465. [9]

Mukherjee, R., Pillai, N. S., and Lin, X. (2015), "Hypothesis Testing for High-Dimensional Sparse Binary Regression," *The Annals of Statistics*, 43, 352–381. [3]

Pennington, J., Socher, R., and Manning, C. D. (2014), "GloVe: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543. [2,9]

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018), "Deep Contextualized Word Representations," in *Proceedings of NAACL*. [13]

R Core Team (2016), *R: A Language and Environment for Statistical Computing*, Vienna, Austria: R Foundation for Statistical Computing. [7]

Randolph, T. W., Zhao, S., Copeland, W., Hullar, M., and Shojaie, A. (2015), "Kernel-Penalized Regression for Analysis of Microbiome Data," *The Annals of Applied Statistics*, 12, 540. [1]

Ridenhour, B. J., Brooker, S. L., Williams, J. E., Van Leuven, J. T., Miller, A. W., Dearing, M. D., and Remien, C. H. (2017), "Modeling Time-Series Data From Microbial Communities," *The ISME Journal*, 11, 2526–2537. [1]

Schloss, P. D., Westcott, S. L., Ryabin, T., Hall, J. R., Hartmann, M., Hollister, E., Lesniewski, R., Oakley, B., Parks, D., Robinson, C., Sahl, J. W., Stres, B., Thallinger, G. G., Van Horn, D., and Weber, C. (2009), "Introducing Mothur: Open-Source, Platform-Independent, Community-Supported Software for Describing and Comparing Microbial Communities," *Applied and Environmental Microbiology*, 75, 7537–7541. [1]

Sculley, D. (2010), "Web-Scale k-Means Clustering," in *Proceedings of the 19th International Conference on World Wide Web*, WWW'10, Association for Computing Machinery, New York, NY, USA, pp. 1177–1178. [13]

She, Y. (2010), "Sparse Regression With Exact Clustering," *Electronic Journal of Statistics*, 4, 1055–1096. [6]

Shi, P., Zhang, A., and Li, H. (2016), "Regression Analysis for Microbiome Compositional Data," *The Annals of Applied Statistics*, 10, 1019–1040. [1]

Tang, Y., Li, M., and Niclolae, D. L. (2016), "Phylogenetic Dirichlet-Multinomial Model for Microbiome Data," arXiv no. 1610.08974. [4]

Thelwall, M., Buckley, K., Paltoglou, G., Cai, D., and Kappas, A. (2010), "Sentiment in Short Strength Detection Informal Text," *Journal of the Association for Information Science and Technology*, 61, 2544–2558. [12]

Tibshirani, R. J. (1996), "Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society*, Series B, 58, 267–288. [2,3]

Tibshirani, R. J., and Taylor, J. (2011), "The Solution Path of the Generalized Lasso," *The Annals of Statistics*, 39, 1335–1371. [5]

Wallace, M. (2007), "Jawbone Java WordNet API." [8]

Wang, H., Lu, Y., and Zhai, C. (2010), "Latent Aspect Rating Analysis on Review Text Data: A Rating Regression Approach," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'10, ACM, New York, NY, USA, pp. 783–792. [1,8]

Wang, J., Shen, X., Sun, Y., and Qu, A. (2016), "Classification With Unstructured Predictors and an Application to Sentiment Analysis," *Journal of the American Statistical Association*, 111, 1242–1253. [12]

Wang, T., and Zhao, H. (2017a), "A Dirichlet-Tree Multinomial Regression Model for Associating Dietary Nutrients With Gut Microorganisms," *Biometrics*, 73, 792–801. [4]

——— (2017b), "Structured Subcomposition Selection in Regression and Its Application to Microbiome Data Analysis," *The Annals of Applied Statistics*, 11, 771–791. [1,5]

Xia, F., Chen, J., Fung, W. K., and Li, H. (2013), "A Logistic Normal Multinomial Regression Model for Microbiome Compositional Data Analysis," *Biometrics*, 69, 1053–1063. [1]

Yu, G., and Liu, Y. (2016), "Sparse Regression Incorporating Graphical Structure Among Predictors," *Journal of the American Statistical Association*, 111, 707–720. [6]

Zhai, J., Kim, J., Knox, K. S., Twigg, H. L., Zhou, H., and Zhou, J. J. (2018), "Variance Component Selection With Applications to Microbiome Taxonomic Data," *Frontiers in Microbiology*, 9, 509. [6]

Zhang, T., Shao, M.-F., and Ye, L. (2012), "454 Pyrosequencing Reveals Bacterial Diversity of Activated Sludge From 14 Sewage Treatment Plants," *The ISME Journal*, 6, 1137–1147. [1]