Advances in Data Analysis and Classification
https://doi.org/10.1007/s11634-022-00530-6

REGULAR ARTICLE

# Proximal methods for sparse optimal scoring and discriminant analysis

**Summer Atkins**[1] · **Gudmundur Einarsson**[2] · **Line Clemmensen**[3] · **Brendan Ames**[4]

Received: 18 September 2019 / Revised: 26 September 2022 / Accepted: 17 November 2022


**Abstract**
Linear discriminant analysis (LDA) is a classical method for dimensionality reduction, where discriminant vectors are sought to project data to a lower dimensional space for optimal separability of classes. Several recent papers have outlined strategies, based on exploiting sparsity of the discriminant vectors, for performing LDA in the high-dimensional setting where the number of features exceeds the number of observations in the data. However, many of these proposed methods lack scalable methods for solution of the underlying optimization problems. We consider an optimization scheme for solving the sparse optimal scoring formulation of LDA based on block coordinate descent. Each iteration of this algorithm requires an update of a scoring vector, which admits an analytic formula, and an update of the corresponding discriminant vector, which requires solution of a convex subproblem; we will propose several variants of this algorithm where the proximal gradient method or the alternating direction method of multipliers is used to solve this subproblem. We show that the per-iteration cost of

Submitted to the editors DATE.

✉ Brendan Ames
bpames@ua.edu

Summer Atkins
srnatkins@lsu.edu

Gudmundur Einarsson
gudmundur.einarsson2@decode.is

Line Clemmensen
lkhc@dtu.dk

1 Department of Mathematics, Louisiana State University, Baton Rouge, LA 70803-4918, USA

2 deCODE Genetics, Reykjavik, Iceland

3 Department of Applied Mathematics and Computer Science, Technical University of Denmark, Building 324, 2800 Kongens Lyngby, Denmark

4 Department of Mathematics, University of Alabama, Box 870350, Tuscaloosa, AL 35487-0350, USA

Published online: 21 December 2022

these methods scales linearly in the dimension of the data provided restricted regularization terms are employed, and cubically in the dimension of the data in the worst case. Furthermore, we establish that when this block coordinate descent framework generates convergent subsequences of iterates, then these subsequences converge to the stationary points of the sparse optimal scoring problem. We demonstrate the effectiveness of our new methods with empirical results for classification of Gaussian data and data sets drawn from benchmarking repositories, including time-series and multispectral X-ray data, and provide `Matlab` and `R` implementations of our optimization schemes.

**Keywords** Sparse discriminant analysis · Optimal scoring · Proximal gradient method · Alternating direction method of multipliers

**Mathematics Subject Classification** 62H30 · 62J12 · 90C26

## 1 Introduction

Sparse discriminant techniques have become popular in the last decade due to their ability to provide increased interpretation as well as predictive performance for high-dimensional problems where few observations are present. These approaches typically build upon successes from sparse linear regression, in particular the LASSO and its variants [see (Hastie et al. 2013), Section 3.4.2 and Hastie et al. (2012)), by augmenting existing schemes for linear discriminant analysis (LDA) with sparsity-inducing regularization terms, such as the $\ell_1$-norm and elastic net.

Thus far, little focus has been put on the optimization strategies of these sparse discriminant methods, nor their computational cost. We propose three novel optimization strategies to obtain discriminant directions in the high-dimensional setting where the number of observations $n$ is much smaller than the ambient dimension $p$ or when features are highly correlated, and analyze the convergence properties of these methods. The methods are proposed for multi-class sparse discriminant analysis using the sparse optimal scoring formulation with elastic net penalty proposed in Clemmensen et al. (2011); adding both the $\ell_1$- and $\ell_2$-norm penalties gives sparse solutions which, in particular, are competitive when high correlations exist in feature space due to the grouping behaviour of the $\ell_2$-norm. The first two strategies are proximal gradient methods based on modification of the (fast) iterative shrinkage algorithm Beck and Teboulle (2009) for linear inverse problems. The third method uses a variant of the alternating direction method of multipliers similar to that proposed in Ames and Hong (2016). We will formally define each of these terms and discuss them in greater detail in Sect. 2. We will see that these heuristics allow efficient classification of high-dimensional data, which was previously impractical using the current state of the art for sparse discriminant analysis. For example, if a diagonal or low-rank Tikhonov regularization term is used and the number of observations is very small relative to $p$, then the per-iteration cost of each of our algorithms is $\mathcal{O}(p)$; that is, the per-iteration cost of our approach scales linearly with the number of features of our data. Finally, we

provide implementations of our algorithms in the form of the `R` package **accSDA**(see Einarsson et al. (2017)) and `Matlab` software.[1]

### 1.1 Existing approaches for sparse LDA

We begin with a brief overview of existing sparse discriminant analysis techniques. Methods such as Witten and Tibshirani (2011), Tibshirani et al. (2003), Fan and Fan (2008) assume independence between the features in the given data. This can lead to poor performance in terms of feature selection as well as predictions, in particular when high correlations exist. Thresholding methods such as Shao et al. (2011), although proven to be asymptotically optimal, ignore the existing multi-linear correlations when thresholding low correlation estimates. Thresholding, furthermore, does not guarantee an invertible correlation matrix, and often pseudo-inverses must be utilized.

For two-class problems, the analysis of Mai and Zou (2013) established an equivalence between the three methods described in Wu et al. (2008), Clemmensen et al. (2011), Ma et al. (2012). These three approaches are formulated as constrained versions of Fisher's discriminant problem, the optimal scoring problem, and a least squares formulation of linear discriminant analysis, respectively. For scaled regularization parameters, Mai and Zou (2013) showed that they all behave asymptotically as Bayes rules. Another two-class sparse linear discriminant method is the linear programming discriminant method proposed in Cai and Liu (2011), which finds an $\ell_1$-norm penalized estimate of the product between covariance matrix and difference in means.

The sparse optimal scoring (SOS) problem was originally formulated in Clemmensen et al. (2011) as a multi-class problem seeking at most $K - 1$ sparse discriminating directions, where $K$ is the number of classes present, whereas Mai and Zou (2013) was formulated for binary problems. This approach builds on earlier work by Hastie et al. (1994, 1995). Mai and Zou later proposed a multi-class sparse discriminant analysis (MSDA) based on the Bayes rule formulation of linear discriminant analysis in Mai et al. (2019). It imposes only the $\ell_1$-norm penalty, whereas the SOS imposes an elastic net penalty ($\ell_1$- plus $\ell_2$-norm). Adding the $\ell_2$-norm can give better predictive performance, in particular when very high correlations exist in data. MSDA, furthermore, finds all discriminative directions at once, whereas SOS finds them sequentially via deflation. A sequential solution can be an advantage if the number of classes is high, and a solution involving only a few directions (the most discriminating ones) is needed. On the other hand, if $K$ is small, finding all directions at once may be advantageous, in order to not propagate errors in a sequential manner. Sparse optimal scoring models based on the group-LASSO are considered in Merchante et al. (2012), Roth and Fischer (2008), while sparse optimal scoring and sparse discriminant analysis has become a popular tool in cognitive neuroscience Grosenick et al. (2008), among many other domains.

Finally, the zero-variance sparse discriminant analysis approach of Ames and Hong (2016) reformulates the sparse discriminant analysis problem as an $\ell_1$-penalized nonconvex optimization problem in order to sequentially identify discriminative directions in the null-space of the pooled within-class scatter matrix. Most relevant for our dis-

[1] Available at http://bpames.people.ua.edu/software.

cussion here is the use of proximal methods to approximately solve the nonconvex optimization problems in Ames and Hong (2016); we will adopt a similar approach for solving the SOS problem.

### 1.2 Contributions

The proposed optimization algorithms are not inherently novel, as they are specializations of existing algorithmic frameworks widely used in sparse regression. However, the application of these proximal algorithms to solve the sparse optimal scoring problem is novel. Moreover, this specialization highlights the strong relationship between the optimal scoring and regression: optimal scoring generalizes regression, in the sense that optimal scoring simultaneously fits both a linear model and a quantitative encoding of categorical class labels.

Further, the proposed sparse optimal scoring heuristics offer a substantial improvement upon the current state of the art for optimal scoring in terms of computational efficiency when the number of predictor variables is large and dense discriminant vectors are desired. Specifically, we show that the computation required for each iteration of the proposed algorithms scales linearly with the number of predictor variables, which yields a potential significant improvement upon the cubic scaling of the least angle regression-based algorithm initially adopted in Clemmensen et al. (2011); our comparison of computational complexity can be found in Sect. 3.4 and "Appendix A".

We performed a detailed empirical analysis of our proposed heuristics for sparse optimal scoring, including comparisons of classification accuracy and computational complexity for simulated data and real-world data sets drawn from the UC Riverside Time Series Archive (Dau et al. 2018, 2019), investigation of convergence phenomena, and scaling tests. These empirical results agree with our theoretical analyses of computational complexity (Sect. 3.4, App. A) and convergence (Sect. 2.2).

### 1.3 Notation

Before we proceed, we first summarize the notation that is used throughout the text. We denote the space of $p$-dimensional vectors by $\mathbf{R}^p$ and the space of $m$ by $n$ real matrices by $\mathbf{R}^{m\times n}$. Bold capital letters, e.g., $\boldsymbol{X}$, will be used to denote matrices, lower-case bold letters, e.g., $\boldsymbol{x}$, $\boldsymbol{\beta}$, denote vectors, and unbolded letters will denote scalars (unless otherwise noted).

We denote the $p$-dimensional all-zeros and all-ones vectors by $\mathbf{0}_p$ and $\mathbf{1}_p$; we omit the subscript $p$ when the dimension is clear by context. We denote the transpose of a matrix $\boldsymbol{X}$ by $\boldsymbol{X}^T$ and the inverse of (nonsingular) $\boldsymbol{X}$ by $\boldsymbol{X}^{-1}$. On the other hand, we use lower-case superscripts to indicate the indices of elements of sequences of vectors, e.g., $\{\boldsymbol{x}^i\}_{i=0}^{\infty} = \boldsymbol{x}^0, \boldsymbol{x}^1, \ldots$ and use subscripts to denote the indices of elements of sequences of scalars, e.g., $\{\alpha_i\}_{i=0}^{\infty} = \alpha_0, \alpha_1, \ldots$. Subscripts will also indicate indices of entries of vectors and matrices; for example, the value in the first row and second column of matrix $\boldsymbol{X}$ will be denoted by $x_{12}$. We will reserve the character $L$ to denote the Lipschitz constant of a given Lipschitz continuous operator $g : \mathbf{R}^p \mapsto \mathbf{R}^m$.

Conversely, we will use the notation $\mathcal{L}_\mu$ to denote the augmented Lagrangian with respect to parameter $\mu$ of a given equally constrained optimization problem; we will also use $\mathcal{L}$ to denote the (unaugmented) Lagrangian function. Finally, we denote the *subdifferential*, i.e., the set of subgradients, of a convex function $f : \mathbf{R}^n \mapsto \mathbf{R}$ at vector $\boldsymbol{x}$ by

$$\partial f(\boldsymbol{x}) = \left\{ \boldsymbol{\phi} \in \mathbf{R}^n : f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \boldsymbol{\phi}^T(\boldsymbol{y} - \boldsymbol{x}) \text{ for all } \boldsymbol{y} \in \operatorname{dom} f \right\}.$$

## 2 An alternating direction method for sparse discriminant analysis

In this section, we describe a block coordinate descent approach for approximately solving the sparse optimal scoring problem for linear discriminant analysis. Proposed in Hastie et al. (1994), the optimal scoring problem recasts linear discriminant analysis as a generalization of linear regression where both the response variable, corresponding to an optimal labeling or scoring of the classes, and linear model parameters, which yield the discriminant vector, are sought. Specifically, suppose that we have the $n \times p$ data matrix $\boldsymbol{X}$, where the rows of $\boldsymbol{X}$ correspond to observations in $\mathbf{R}^p$ sampled from one of $K$ classes; we assume that the data has been centered so that the sample mean is the zero vector $\mathbf{0} \in \mathbf{R}^p$.

Optimal scoring generates a sequence of discriminant vectors and conjugate scoring vectors as follows. Suppose that we have identified the first $j-1$ discriminant vectors $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{j-1} \in \mathbf{R}^p$ and scoring vectors $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{j-1} \in \mathbf{R}^K$. To calculate the $j$th discriminant vector $\boldsymbol{\beta}_j$ and scoring vector $\boldsymbol{\theta}_j$, we solve the optimal scoring criterion problem

$$\begin{aligned} \underset{\boldsymbol{\theta} \in \mathbf{R}^K,\, \boldsymbol{\beta} \in \mathbf{R}^p}{\arg\min} \quad & \|\boldsymbol{Y}\boldsymbol{\theta} - \boldsymbol{X}\boldsymbol{\beta}\|^2 \\ \text{s.t. } & \tfrac{1}{n}\boldsymbol{\theta}^T\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{\theta} = 1, \quad \boldsymbol{\theta}^T\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{\theta}_\ell = 0 \ \ \forall \ell < j, \end{aligned} \tag{1}$$

where $\boldsymbol{Y}$ denotes the $n \times K$ indicator matrix for class membership, defined by $y_{\ell m} = 1$ if the $\ell$th observation belongs to the $m$th class, and $y_{\ell m} = 0$ otherwise, and $\|\cdot\| : \mathbf{R}^n \to \mathbf{R}$ denotes the vector $\ell_2$-norm on $\mathbf{R}^n$ defined by $\|\boldsymbol{y}\| = \sqrt{y_1^2 + y_2^2 + \cdots + y_n^2}$ for all $\boldsymbol{y} \in \mathbf{R}^n$. We direct the reader to Hastie et al. (1994) for further details regarding the derivation of (1).

A variant of the optimal scoring problem called *sparse discriminant analysis* or *sparse optimal scoring*, which employs regularization via the elastic net penalty function is proposed in Clemmensen et al. (2011). As before, suppose that we have identified the first $j-1$ discriminant vectors $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{j-1}$ and scoring vectors $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{j-1}$. We calculate the $j$th sparse discriminant vector $\boldsymbol{\beta}_j$ and scoring vector $\boldsymbol{\theta}_j$ as the optimal solutions of the optimal scoring criterion problem

$$\begin{aligned} \underset{\boldsymbol{\theta} \in \mathbf{R}^K,\, \boldsymbol{\beta} \in \mathbf{R}^p}{\arg\min} \quad & \|\boldsymbol{Y}\boldsymbol{\theta} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \gamma\boldsymbol{\beta}^T\boldsymbol{\Omega}\boldsymbol{\beta} + \lambda\|\boldsymbol{\beta}\|_1 \\ \text{s.t. } & \tfrac{1}{n}\boldsymbol{\theta}^T\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{\theta} = 1, \quad \boldsymbol{\theta}^T\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{\theta}_\ell = 0 \ \ \forall \ell < j, \end{aligned} \tag{2}$$

where $\|\cdot\|_1 : \mathbf{R}^p \to \mathbf{R}$ denotes the vector $\ell_1$-norm on $\mathbf{R}^p$ defined by $\|\boldsymbol{x}\|_1 = |x_1| + |x_2| + \cdots + |x_p|$ for all $\boldsymbol{x} \in \mathbf{R}^p$, $\boldsymbol{Y} \in \mathbf{R}^{n\times K}$ is again the indicator matrix for class membership, $\lambda$ and $\gamma$ are nonnegative tuning parameters, and $\boldsymbol{\Omega}$ is a $p \times p$ positive definite matrix. That is, (2) is the result of adding regularization to the optimal scoring problem using a linear combination of the Tikhonov penalty term $\boldsymbol{\beta}^T\boldsymbol{\Omega}\boldsymbol{\beta}$ and the $\ell_1$-norm penalty $\|\boldsymbol{\beta}\|_1$; we will provide further discussion regarding the choice of $\boldsymbol{\Omega}$ in Sect. 3.4. The optimization problem (2) is nonconvex, due to the presence of nonconvex spherical constraints. As such, we do not expect to find a globally optimal solution of (2) using iterative methods.

### 2.1 Block coordinate descent for sparse optimal scoring

Clemmensen et al. propose a block coordinate descent method to solve (2) in Clemmensen et al. (2011). This approach has been widely adopted, with nearly 600 citations according to Google Scholar,[2] and over 119000 downloads of the `R` implementation `sparseLDA` from the Comprehensive R Archive Network (CRAN).[3] This approach can be described as follows. Suppose that we have an estimate $(\boldsymbol{\theta}^i, \boldsymbol{\beta}^i)$ of $(\boldsymbol{\theta}_j, \boldsymbol{\beta}_j)$ after $i$ iterations. To update $\boldsymbol{\theta}^i$, we fix $\boldsymbol{\beta} = \boldsymbol{\beta}^i$ and solve the optimization problem

$$\begin{aligned}\boldsymbol{\theta}^{i+1} = \underset{\boldsymbol{\theta}\in\mathbf{R}^K}{\arg\min}\ & \|\boldsymbol{Y}\boldsymbol{\theta} - \boldsymbol{X}\boldsymbol{\beta}^i\|^2 \\ \text{s.t. } & \tfrac{1}{n}\boldsymbol{\theta}^T\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{\theta} = 1,\ \ \boldsymbol{\theta}^T\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{\theta}_\ell = 0\ \ \forall \ell < j.\end{aligned} \tag{3}$$

The subproblem (3) is nonconvex in $\boldsymbol{\theta}$, however, it is known that (3) admits an analytic solution and can be solved exactly in polynomial time. Indeed, we have the following lemma providing an analytic update formula for $\boldsymbol{\theta}$. Note that this update requires $\mathcal{O}(K^3 + pn)$ floating point operations to perform the necessary matrix products. For completeness, we provide a proof of Lemma 1 in "Appendix B". See (Clemmensen et al. 2011, Section 2.2) for more details.

**Lemma 1** *The problem* (3) *has optimal solution*

$$\boldsymbol{\theta}^{i+1} = s\left(\boldsymbol{I} - \frac{1}{n}\boldsymbol{Q}_j\boldsymbol{Q}_j^T\boldsymbol{Y}^T\boldsymbol{Y}\right)(\boldsymbol{Y}^T\boldsymbol{Y})^{-1}\boldsymbol{Y}^T\boldsymbol{X}\boldsymbol{\beta}^i, \tag{4}$$

*where* $\boldsymbol{Q}_j$ *is the* $K \times j$ *matrix with columns consisting of the* $j-1$ *scoring vectors* $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{j-1}$ *and the all-ones vector* $\mathbf{1} \in \mathbf{R}^K$*, and* $s$ *is a proportionality constant ensuring that* $(\boldsymbol{\theta}^{i+1})^T\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{\theta}^{i+1} = n$*. In particular,* $\boldsymbol{\theta}^{i+1}$ *is given by*

$$\boldsymbol{w} = \left(\boldsymbol{I} - \frac{1}{n}\boldsymbol{Q}_j\boldsymbol{Q}_j^T\boldsymbol{Y}^T\boldsymbol{Y}\right)(\boldsymbol{Y}^T\boldsymbol{Y})^{-1}\boldsymbol{Y}^T\boldsymbol{X}\boldsymbol{\beta}^i, \qquad \boldsymbol{\theta}^{i+1} = \frac{\sqrt{n}\boldsymbol{w}}{\|\boldsymbol{Y}\boldsymbol{w}\|}. \tag{5}$$

[2] Citation count available from scholar.google.com accessed November 11, 2021.

[3] Download count available from cranlogs.r-pkg.org/badges/grand-total/sparseLDA accessed November 11, 2021.

After we have updated $\boldsymbol{\theta}^{i+1}$, we obtain $\boldsymbol{\beta}^{i+1}$ by solving the unconstrained optimization problem

$$\boldsymbol{\beta}^{i+1} = \underset{\boldsymbol{\beta}\in\mathbf{R}^p}{\arg\min} \|\boldsymbol{Y}\boldsymbol{\theta}^{i+1} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \gamma\boldsymbol{\beta}^T\boldsymbol{\Omega}\boldsymbol{\beta} + \lambda\|\boldsymbol{\beta}\|_1. \tag{6}$$

That is, we update $\boldsymbol{\beta}^{i+1}$ by solving the generalized elastic net problem (6). We stop this block update scheme if the relative change in consecutive iterates is smaller than a desired tolerance. Specifically, we stop the algorithm if

$$\max\left\{\frac{\|\boldsymbol{\theta}^{i+1} - \boldsymbol{\theta}^i\|}{\|\boldsymbol{\theta}^{i+1}\|}, \frac{\|\boldsymbol{\beta}^{i+1} - \boldsymbol{\beta}^i\|}{\|\boldsymbol{\beta}^{i+1}\|}\right\} \leq \epsilon$$

for stopping tolerance $\epsilon > 0$. We delay discussion of strategies until after a discussion of the convergene properties of this alternating minimization scheme. Specifically, we discuss an algorithm based on least-angle regression (LARS) for solving (6) suggested in Clemmensen et al. (2011) in Sect. 2.3 and our proposed improvements in Sect. 3.

### 2.2 Convergence of the block coordinate descent algorithm

Before we proceed to the statement of our algorithms for updating $\boldsymbol{\beta}$, we investigate the convergence properties of the block coordinate descent method given by Algorithm 1. Our two main results, Theorems 1 and 2, are specializations of standard results for alternating minimization algorithms; we provide proofs of these results as appendices. We should note that these two theorems establish convergence properties of Algorithm 1 that are independent of the algorithm used to solve Subproblem (6). In particular, these convergence theorems hold if we use any of the iterative methods suggested in the following section for solving (6), as well as the LARS Algorithm for solving (6) suggested in Clemmensen et al. (2011).

We first note that the Lagrangian $\mathcal{L} : \mathbf{R}^K \times \mathbf{R}^p \times \mathbf{R} \times \mathbf{R}^{j-1} \to \mathbf{R}$ of (2) is given by

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\beta}, \psi, \boldsymbol{v}) =& \|\boldsymbol{Y}\boldsymbol{\theta} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \gamma\boldsymbol{\beta}^T\boldsymbol{\Omega}\boldsymbol{\beta} + \lambda\|\boldsymbol{\beta}\|_1 \\ &+ \psi(\boldsymbol{\theta}^T\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{\theta} - n) + \boldsymbol{v}^T\boldsymbol{U}\boldsymbol{\theta},\end{aligned}$$

where $\boldsymbol{U}^T = (\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{\theta}_1, \boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{\theta}_2, \ldots, \boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{\theta}_{j-1})$. Note that the Lagrangian is *not* a convex function in general. However, $\mathcal{L}$ is the sum of the (possibly) nonconvex quadratic $\|\boldsymbol{Y}\boldsymbol{\theta} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \psi(\boldsymbol{\theta}^T\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{\theta} - n) + \gamma\boldsymbol{\beta}^T\boldsymbol{\Omega}\boldsymbol{\beta} + \boldsymbol{v}^T\boldsymbol{U}\boldsymbol{\theta}$ and the convex nonsmooth function $\lambda\|\boldsymbol{\beta}\|_1$; therefore, $\mathcal{L}$ is subdifferentiable, with subdifferential at $(\boldsymbol{\beta}, \boldsymbol{\theta})$ given by the sum of the gradient of the smooth term at $(\boldsymbol{\beta}, \boldsymbol{\theta})$ and the subdifferential of the convex nonsmooth term at $(\boldsymbol{\beta}, \boldsymbol{\theta})$.

We now provide our first convergence result, specifically, that Algorithm 1 generates a convergent sequence of function values.

**Theorem 1** *Suppose that the sequence of iterates* $\{(\boldsymbol{\theta}^i, \boldsymbol{\beta}^i)\}_{i=0}^{\infty}$ *is generated by Algorithm 1. Then the sequence of objective function values* $\{F(\boldsymbol{\theta}^i, \boldsymbol{\beta}^i)\}_{i=0}^{\infty}$ *defined by* $F(\boldsymbol{\theta}, \boldsymbol{\beta}) := \|\boldsymbol{Y}\boldsymbol{\theta} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \gamma \boldsymbol{\beta}^T \boldsymbol{\Omega} \boldsymbol{\beta} + \lambda \|\boldsymbol{\beta}\|_1$ *is convergent.*

We include a proof of Theorem 1 in "Appendix C".

We also have the following theorem, which establishes that every convergent subsequence of $\{(\boldsymbol{\theta}^i, \boldsymbol{\beta}^i)\}_{i=1}^{\infty}$ converges to a stationary point of (2).

**Theorem 2** *Let* $\{(\boldsymbol{\theta}^i, \boldsymbol{\beta}^i)\}_{i=1}^{\infty}$ *be the sequence of points generated by Algorithm 1. If* $\{(\boldsymbol{\theta}^{i_\ell}, \boldsymbol{\beta}^{i_\ell})\}_{\ell=1}^{\infty}$ *is a convergent subsequence of* $\{(\boldsymbol{\theta}^i, \boldsymbol{\beta}^i)\}_{i=1}^{\infty}$ *with limit* $(\boldsymbol{\theta}^*, \boldsymbol{\beta}^*)$ *then* $(\boldsymbol{\theta}^*, \boldsymbol{\beta}^*)$ *is a* stationary point *of* (2)*:* $(\boldsymbol{\theta}^*, \boldsymbol{\beta}^*)$ *is feasible for* (2) *and there exists* $\psi^* \in \mathbf{R}$ *and* $\boldsymbol{v}^* \in \mathbf{R}^{j-1}$ *such that* $\mathbf{0} \in \partial \mathcal{L}(\boldsymbol{\theta}^*, \boldsymbol{\beta}^*, \psi^*, \boldsymbol{v}^*)$*, where* $\partial \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\beta}, \psi, \boldsymbol{v})$ *denotes the subdifferential of the Lagrangian function* $\mathcal{L}$ *with respect to the primal variables* $(\boldsymbol{\theta}, \boldsymbol{\beta})$.

A proof of Theorem 2 can be found in "Appendix D".

We conclude this section by noting that we expect the block coordinate descent method to converge after exactly one full iteration in the absence of rounding error in the two-class case, i.e., $K = 2$. In this case, the optimal solution of (3) is given by the projection of any vector $\boldsymbol{\theta}$ not in the span of the all-ones vector $\mathbf{1}$ onto the set

$$\left\{\boldsymbol{\theta} \in \mathbf{R}^k : \boldsymbol{\theta}^T \boldsymbol{Y}^T \boldsymbol{Y} \boldsymbol{\theta} = n,\ \boldsymbol{\theta}^T \boldsymbol{Y}^T \boldsymbol{Y} \mathbf{1} = 0\right\};$$

this is equivalent to the optimal solution given by Lemma 1. This solution is uniquely defined (up to sign) and is obtained after the initial $\boldsymbol{\theta}$ update if the initial solution $\boldsymbol{\theta}$ is not a scalar multiple of $\mathbf{1}$. This suggests that Algorithm 1 will converge after exactly one iteration if we solve (6) exactly. In practice, we may require multiple iterations of Algorithm 1 depending on the relative stopping tolerances of Algorithm 1 and the method used to solve (6).

### 2.3 The least angle regression algorithm

It is suggested in Clemmensen et al. (2011) that (6) can be solved using the *least angle regression (LARS-EN)* algorithm proposed in Zou and Hastie (2005) for solving elastic net regularized linear inverse problems, which in turn generalizes the LARS algorithm for $\ell_1$ regularized linear inverse problems proposed in Efron et al. (2004). This method sequentially updates elastic net estimates of the solution of (6). The main computational step of the $i$th iteration is the inversion of a coefficient matrix of the form $\boldsymbol{X}_{A_i}^T \boldsymbol{X}_{A_i} + \gamma \boldsymbol{\Omega}$, where $A_i$ is the active variable set, i.e, the indices of nonzero entries of the $i$th iterate $\boldsymbol{\beta}^i$. To minimize computational costs, a low-rank dow- ndating procedure is used to update the Cholesky factorization of $\boldsymbol{X}_{A_{i-1}}^T \boldsymbol{X}_{A_{i-1}} + \gamma \boldsymbol{\Omega}$ and only nonzero-cofficients and active variable sets are stored each iteration. The specialization of the LARS algorithm for solution of (6) is included in the `Matlab` and `R` package sparseLDA Clemmensen (2008).

It is known that the number of iterations of the LARS algorithm acts as a tuning parameter for sparsity of solution (induced by the $\ell_1$ penalty in (6)). That is, the

**Algorithm 1:** Block Coordinate Descent for SDA (2)

**Data**: Given stopping tolerance $\epsilon$, and maximum number of iterations $N$.
**Result**: Discriminant vectors $(\boldsymbol{\theta}_1^*, \boldsymbol{\beta}_1^*)$, $(\boldsymbol{\theta}_2^*, \boldsymbol{\beta}_2^*)$, …, $(\boldsymbol{\theta}_{K-1}^*, \boldsymbol{\beta}_{K-1}^*)$ calculated as approximate solutions of (2).
**for** $j = 1, 2, \ldots, K-1$

Initialize $\boldsymbol{\theta}_j^0$ as the projection of $K$-dimensional vector $\boldsymbol{z}$ with entries sampled uniformly at random from the interval [0, 1] onto the feasible region using the identity

$$\boldsymbol{\theta}_j^0 = \left(\boldsymbol{I} - \frac{1}{n}\boldsymbol{Q}_j\boldsymbol{Q}_j^T\boldsymbol{Y}^T\boldsymbol{Y}\right)(\boldsymbol{Y}^T\boldsymbol{Y})^{-1}\boldsymbol{z}, \qquad \boldsymbol{\theta}_j^0 = \frac{\sqrt{n}\boldsymbol{\theta}_j^0}{\|\boldsymbol{Y}\boldsymbol{\theta}_j^0\|};$$

Calculate the $j$th scoring and discriminant vector pair $(\boldsymbol{\theta}_j^*, \boldsymbol{\beta}_j^*)$ as the limit point of the sequence $\{(\boldsymbol{\theta}_j^i, \boldsymbol{\beta}_j^i)\}_{i=0}^{\infty}$ as follows:
**for** $i = 0, 1, 2 \ldots N$

Update $\boldsymbol{\beta}_j^i$ as the solution of (6) with $\boldsymbol{\theta} = \boldsymbol{\theta}_j^i$ using the solution returned by one of (2), (4), (6), and (5);
Update $\boldsymbol{\theta}_j^{i+1}$ by

$$\boldsymbol{w} = \left(\boldsymbol{I} - \frac{1}{n}\boldsymbol{Q}_j\boldsymbol{Q}_j^T\boldsymbol{Y}^T\boldsymbol{Y}\right)(\boldsymbol{Y}^T\boldsymbol{Y})^{-1}\boldsymbol{Y}^T\boldsymbol{X}\boldsymbol{\beta}_j^i, \qquad \boldsymbol{\theta}^{i+1} = \frac{\sqrt{n}\boldsymbol{w}}{\|\boldsymbol{Y}\boldsymbol{w}\|};$$

Declare convergence if the residual between consecutive iterates is smaller than desired tolerance:
**if** $\max\left\{\frac{\|\boldsymbol{\theta}^{i+1}-\boldsymbol{\theta}^i\|}{\|\boldsymbol{\theta}^{i+1}\|}, \frac{\|\boldsymbol{\beta}^{i+1}-\boldsymbol{\beta}^i\|}{\|\boldsymbol{\beta}^{i+1}\|}\right\} < \epsilon$

The algorithm has converged;
**break**;

**end**

**end**

**end**

sequence of iterates generated by the LARS algorithm is equivalent to set of solutions of (6) for a particular sequence of choices of regularization parameter $\lambda$. This yields two potential improvements over other candidate methods for solving (6). First, we do not need to repeatedly solve (6) to tune the parameter $\lambda$, e.g., as part of a cross validation or boosting scheme. Instead, we can obtain a full regularization path with a single call to the LARS algorithm. Second, in many applications, it is more natural to seek a solution with a specific maximum cardinality, rather than some desired value of $\ell_1$ penalty; the LARS algorithm allows the use of this more natural interpretation of the regularization process as a stopping criterion. We direct the reader to (Zou and Hastie 2005, Section 3.5) for further details.

This approach carries a computational cost on the order of $\mathcal{O}(mnp + m^3)$, where $m$ is the desired number of nonzero coefficients at termination, which is prohibitively expensive if both $p$ and $m$ are large. For example, if $m = cp$ for some constant $c \in (0, 1)$, then the per-iteration cost scales cubically with $p$. Moreover, we should note that we solve a different instance of (6) during each iteration of Algorithm 1, since the value of $\boldsymbol{\theta}$ changes each iteration. Thus, we are required to compute the full

regularization path during each iteration of Algorithm 1. We should also note that the solutions given by LARS-based algorithms are not directly comparable to those given by proximal gradient methods since they do not correspond to the same optimization problem. Generally, the deflationary process for calculating discriminant-scoring vector pairs leads to different optimization problems for calculating $(\boldsymbol{\theta}_k, \boldsymbol{\beta}_k)$ for $k > 1$, due to differences in results of solving the prior subproblems between heuristics. This phenomena is especially pronounced when using LARS-based algorithms, compared to our proposed proximal methods, since the LARS-based algorithms implicitly use dynamically updated regularization parameters $\lambda_i$, which inherently depend on the choice of discriminant and scoring vectors chosen in earlier steps of Algorithm 1.

Here, we must be careful to note that the optimization problem is well-defined in the form (1) for each step of Algorithm 1 *in the absence of error*. That is, if we solve each subproblem for $(\boldsymbol{\theta}_i^*, \boldsymbol{\beta}_i^*)$, $i = 1, 2, \ldots, K-1$ *exactly*, then we should obtain the same set of scoring-discriminant vector pairs $(\boldsymbol{\theta}_i^*, \boldsymbol{\beta}_i^*)$, $i = 1, 2, \ldots, K-1$ regardless of which method we use to solve (1). However, in practice, we terminate our heuristics for solving the subproblem (1) for calculating $(\boldsymbol{\theta}_i^*, \boldsymbol{\beta}_i^*)$ after a desired level of suboptimality is met, which results in different approximate solutions using different heuristics for solving (1). This, in turn, gives different instances of (1) for calculating $(\boldsymbol{\theta}_{i+1}^*, \boldsymbol{\beta}_{i+1}^*)$, which is compounded as we solve each subproblem for $i = 1, 2, \ldots, K-1$.

Finally, we note that coordinate descent methods have been widely adopted for calculation of elastic net regularized generalized linear models; see Friedman et al. (2010) for further details. However, we are unaware of any application of coordinate descent methods for solution of the elastic net regularized optimal scoring problem (2).

## 3 Proximal methods for updating $\boldsymbol{\beta}$

Our primary contribution is a collection of algorithms for solving the $\boldsymbol{\beta}$-update subproblem (6). Specifically, we specialize three classical algorithms, each based on the evaluation of proximal operators, to obtain novel numerical methods for solution of the sparse optimal scoring problem. We will see that these algorithms require significantly fewer computational resources than least angle regression if we exploit structure in the regularization parameter $\boldsymbol{\Omega}$.

### 3.1 The proximal gradient algorithm

Given a convex function $f : \mathbf{R}^p \to \mathbf{R}$, the *proximal operator* $\operatorname{prox}_f : \mathbf{R}^p \to \mathbf{R}^p$ of $f$ is defined by

$$\operatorname*{prox}_f(\boldsymbol{y}) = \operatorname*{arg\,min}_{\boldsymbol{x} \in \mathbf{R}^p} \left\{ f(\boldsymbol{x}) + \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2 \right\},$$

which yields a point that balances the competing objectives of being near $\boldsymbol{y}$ while simultaneously minimizing $f$. The use of proximal operators is a classical technique

in optimization, particularly as surrogates for gradient descent steps for minimization of nonsmooth functions. For example, consider the optimization problem

$$\min_{\boldsymbol{x}\in\mathbf{R}^p} f(\boldsymbol{x}) + g(\boldsymbol{x}), \tag{7}$$

where $f : \mathbf{R}^p \to \mathbf{R}$ is differentiable and $g : \mathbf{R}^p \to \mathbf{R}$ is potentially nonsmooth. That is, (7) minimizes an objective that can be decomposed as the sum of a differentiable function $f$ and nonsmooth function $g$. To solve (7), the *proximal gradient method* performs iterations consisting of a step in the direction of the negative gradient $-\nabla f$ of the smooth part $f$ followed by evaluation of the proximal operator of $g$: given iterate $\boldsymbol{x}^i$, we obtain the updated iterate $\boldsymbol{x}^{i+1}$ by

$$\boldsymbol{x}^{i+1} = \operatorname*{prox}_{\alpha_i g}(\boldsymbol{x}^i - \alpha_i \nabla f(\boldsymbol{x}^i)), \tag{8}$$

where $\alpha_i$ is a step length parameter. If both $f$ and $g$ are differentiable and the step size $\alpha_i$ is small, then this approach reduces to the classical gradient descent iteration: $\boldsymbol{x}^{i+1} \approx \boldsymbol{x}^i - \alpha_i \nabla f(\boldsymbol{x}^i) - \alpha_i \nabla g(\boldsymbol{x}^i)$. We direct the reader to Beck and Teboulle (2009), Parikh and Boyd (2014) for more details regarding the proximal gradient method.

Expanding the residual norm term $\|\boldsymbol{Y\theta} - \boldsymbol{X\beta}\|^2$ in the objective of (6) and dropping the constant term shows that (6) is equivalent to minimizing

$$F(\boldsymbol{\beta}) = \frac{1}{2}\boldsymbol{\beta}^T \boldsymbol{A\beta} + \boldsymbol{d}^T\boldsymbol{\beta} + \lambda\|\boldsymbol{\beta}\|_1, \tag{9}$$

where $\boldsymbol{A} = 2(\boldsymbol{X}^T\boldsymbol{X} + \gamma\boldsymbol{\Omega})$ and $\boldsymbol{d} = -2\boldsymbol{X}^T\boldsymbol{Y\theta}^{i+1}$. We can decompose $F$ as $F(\boldsymbol{\beta}) = f(\boldsymbol{\beta}) + g(\boldsymbol{\beta})$, where $f(\boldsymbol{\beta}) = \frac{1}{2}\boldsymbol{\beta}^T\boldsymbol{A\beta} + \boldsymbol{d}^T\boldsymbol{\beta}$ and $g(\boldsymbol{\beta}) = \lambda\|\boldsymbol{\beta}\|_1$. Note that $F$ is strongly convex if the penalty matrix $\boldsymbol{\Omega}$ is positive definite; in this case (9) has a unique minimizer. Note further that $f$ is differentiable with $\nabla f(\boldsymbol{\beta}) = \boldsymbol{A\beta} + \boldsymbol{d}$. Moreover, the proximal operator of the $\ell_1$-norm term $g(\boldsymbol{\beta}) = \lambda\|\boldsymbol{\beta}\|_1$ is given by

$$\operatorname*{prox}_{\lambda\|\cdot\|_1}(\boldsymbol{y}) = \operatorname{sign}(\boldsymbol{y})\max\{|\boldsymbol{y}| - \lambda\mathbf{1}, \mathbf{0}\} =: S_\lambda(\boldsymbol{y});$$

see (Parikh and Boyd 2014, Section 6.5.2). The proximal operator $S_\lambda = \operatorname{prox}_{\lambda\|\cdot\|_1}$ is often called the *soft thresholding operator* (with respect to the threshold $\lambda$) and $\operatorname{sign} : \mathbf{R}^p \to \mathbf{R}^p$ and $\max : \mathbf{R}^p \times \mathbf{R}^p \to \mathbf{R}^p$ are the element-wise sign and maximum mappings defined by

$$[\operatorname{sign}(\boldsymbol{y})]_i = \operatorname{sign}(y_i) = \begin{cases} +1, \text{ if } y_i > 0 \\ \ \ 0, \text{ if } y_i = 0 \\ -1, \text{ if } y_i < 0 \end{cases}$$

and $[\max(\boldsymbol{x}, \boldsymbol{y})]_i = \max(x_i, y_i)$. Using this decomposition, we can apply the proximal gradient method to generate a sequence of iterates $\{\boldsymbol{\beta}^i\}$ by

$$\boldsymbol{\beta}^{i+1} = \operatorname{sign}(\boldsymbol{p}^i)\max\{|\boldsymbol{p}^i| - \lambda\alpha_i\mathbf{1}, \mathbf{0}\}, \tag{10}$$

**Algorithm 2:** Proximal gradient method for solving (6)

**Data**: Given initial iterate $\boldsymbol{\beta}^0$, sequence of step sizes $\{\alpha_i\}_{i=0}^{\infty}$, stopping tolerance $\epsilon$, and maximum number of iterations $N$.
**Result**: Solution $\boldsymbol{\beta}^*$ of (6).
**for** $i = 0, 1, 2 \ldots N$
  Update gradient term by (11):

$$\boldsymbol{p}^i = \boldsymbol{\beta}^i - \alpha_i(\boldsymbol{A}\boldsymbol{\beta}^i + \boldsymbol{d});$$

  Update iterate using proximal gradient step (10):

$$\boldsymbol{\beta}^{i+1} = \text{sign}(\boldsymbol{p}^i)\max\{|\boldsymbol{p}^i| - \lambda\alpha_i\mathbf{1}, \mathbf{0}\};$$

  Terminate if current iterate is approximately stationary:
  **if** $\|\boldsymbol{A}\boldsymbol{\beta}^{i+1} + \boldsymbol{d} + \lambda\,\text{sign}(\boldsymbol{\beta}^{i+1})\|_\infty < p\epsilon$:
    The algorithm has converged;
    **break**;
  **end**
**end**

where

$$\boldsymbol{p}^i = \boldsymbol{\beta}^i - \alpha_i\nabla f(\boldsymbol{\beta}^i) = \boldsymbol{\beta}^i - \alpha_i(\boldsymbol{A}\boldsymbol{\beta}^i + \boldsymbol{d}); \tag{11}$$

here, $\mathbf{1}$ and $\mathbf{0}$ denote the all-ones and all-zeros vectors in $\mathbf{R}^p$, respectively. This proximal gradient algorithm with constant step lengths is summarized in Algorithm 2; Algorithm 3 can be modified to obtain a variant of Algorithm 2 that employs a backtracking line search. It is important to note that this update scheme is virtually identical to the *iterative soft thresholding algorithm* (ISTA) proposed in Beck and Teboulle (2009) for solving $\ell_1$ regularized linear inverse problems. Specifically, our problem and update formula differs only from that typically associated with ISTA in the presence of the Tikhonov regression term $\boldsymbol{\beta}^T\boldsymbol{\Omega}\boldsymbol{\beta}$ in our model. As an immediate consequence, we can specialize the known convergence properties of ISTA (Beck and Teboulle 2009, Theorem 3.1) to show that the sequence of function values $\{F(\boldsymbol{\beta}^i)\}$ generated by Algorithm 2 converges sublinearly to the optimal function value of (9) at a rate no worse than $\mathcal{O}(1/i)$ for a certain choice of step lengths $\{\alpha_i\}$.

**Theorem 3** (Specialization of (Beck and Teboulle 2009, Theorem 3.1)) *Let $\{\boldsymbol{\beta}^i\}$ be generated by Algorithm 2 with initial iterate $\boldsymbol{\beta}^0$ and constant step size $\alpha_i = \alpha \in (0, 1/\|\boldsymbol{A}\|)$ or step sizes chosen using the backtracking scheme given by Algorithm 3, where $\|\boldsymbol{A}\| = \lambda_{\max}(\boldsymbol{A})$ denotes the largest eigenvalue of the positive semidefinite matrix $\boldsymbol{A}$. Suppose that $\boldsymbol{\beta}^*$ is a minimizer of $F$. Then there exists a constant $c$ such that*

$$F(\boldsymbol{\beta}^i) - F(\boldsymbol{\beta}^*) \leq \frac{c\|\boldsymbol{A}\|\|\boldsymbol{\beta}^0 - \boldsymbol{\beta}^*\|^2}{i} \tag{12}$$

*for any $i \geq 1$.*

**Algorithm 3:** Proximal gradient method for solving (6) with back tracking line search

**Data**: Given initial iterate $\boldsymbol{\beta}^0$, sequence of step sizes $\{\alpha_i\}_{i=0}^{\infty}$, stopping tolerance $\epsilon$, and maximum number of iterations $N$.
**Result**: Solution $\boldsymbol{\beta}^*$ of (6).
**for** $i = 0, 1, 2 \ldots N$
  Update iterate using proximal gradient step (10) and backtracking line search:
  **for** $k = 0, 1, 2 \ldots$ until step size accepted
    Update step length:

$$\bar{L} = \eta^k L_i, \qquad \alpha = \frac{1}{\bar{L}};$$

    Update iterate using proximal gradient step (10):

$$\boldsymbol{p}^i = \boldsymbol{\beta}^i - \alpha_i(\boldsymbol{A}\boldsymbol{\beta}^i + \boldsymbol{d});$$
$$\boldsymbol{\beta}^{i+1} = \text{sign}(\boldsymbol{p}^{i+1}) \max\{|\boldsymbol{p}^{i+1}| - \lambda\alpha\mathbf{1}, \mathbf{0}\};$$

    Determine whether to accept update or increment step length:
    **if** $(\boldsymbol{\beta}^{i+1} - \boldsymbol{y}^{i+1})^T \left(\frac{\bar{L}}{2}\boldsymbol{I} - \boldsymbol{A}\right)(\boldsymbol{\beta}^{i+1} - \boldsymbol{y}^{i+1}) \geq 0$
      Accept update: set $L_{i+1} = \bar{L}$ and $\alpha_i = \bar{\alpha}$;
      **break**;
    **end**
  **end**
  Terminate if current iterate is approximately stationary:
  **if** $\|\boldsymbol{A}\boldsymbol{\beta}^{i+1} + \boldsymbol{d} + \lambda\,\text{sign}(\boldsymbol{\beta}^{i+1})\|_\infty < p\epsilon$:
    The algorithm has converged;
    **break**;
  **end**
**end**

Note that Theorem 3 implies that Algorithms 2 and 3 yield $\epsilon$-suboptimal solutions for (6) within $\mathcal{O}(1/\epsilon)$ iterations. Here, we say that a vector $\tilde{\boldsymbol{\beta}}$ is $\epsilon$-suboptimal for (6) if $F(\tilde{\boldsymbol{\beta}}) - F(\boldsymbol{\beta}^*) \leq \epsilon$, for each $\epsilon > 0$. It is known that ISTA converges *linearly* when the objective function $F$ is strongly convex (see (Beck 2017, Chapter 10)). We will see that the strong convexity of the objective of (2) depends on the structure of the regularization term $\boldsymbol{\Omega}$. When $\boldsymbol{\Omega}$ is full rank, then $F$ is strongly convex. Therefore, the sequence of iterates generated by Algorithm 2 converges to the unique minimizer of (6) if the penalty parameter $\boldsymbol{\Omega}$ is chosen to be positive definite. If we choose $\boldsymbol{\Omega}$ to be positive semidefinite but not full rank, then $F$ may not be strongly convex. In this case, Theorem 3 establishes that the sequence of iterates generated by Algorithm 2 converges sublinearly to the minimum value of (6) and any limit point of this sequence is a minimizer of (6). We will see that using such a matrix may have attractive computational advantages despite this loss of uniqueness.

It is reasonably easy to see that the quadratic term of $F$ in (9) is differentiable and has Lipschitz continuous gradient with constant $L = \|\boldsymbol{A}\|$; this is the significance of the $\|\boldsymbol{A}\|$ term in (12). In order to ensure convergence in our proximal gradient method, we need to estimate $\|\boldsymbol{A}\|$ to choose a sufficiently small step size $\alpha$. Computing this Lipschitz constant may be prohibitively expensive for large $p$; one can typically calculate $\|\boldsymbol{A}\|$ to arbitrary precision using variants of the Power Method (see Golub and

Van Loan 2013, Sections 7.3.1, 8.2) at a cost of $\mathcal{O}(p^2 \log p)$ floating point operations. Instead, we could use an upper bound $\tilde{L} \geq L$ to compute our constant step size $\alpha = 1/\tilde{L} \leq 1/L$. For example, when $\boldsymbol{\Omega}$ is a diagonal matrix, we estimate $\|\boldsymbol{A}\|$ by

$$\|\boldsymbol{A}\| = 2\|\gamma\boldsymbol{\Omega} + \boldsymbol{X}^T\boldsymbol{X}\| \leq 2\gamma\|\text{diag}(\boldsymbol{\Omega})\|_\infty + 2\|\boldsymbol{X}\|_F^2 \approx \frac{1}{\alpha},$$

where $\text{diag}(\boldsymbol{M}) \in \mathbf{R}^p$ is the vector of diagonal entries of the matrix $\boldsymbol{M} \in \mathbf{R}^{p\times p}$. Here, we used the triangle inequality and the identity $\|\boldsymbol{X}^T\boldsymbol{X}\| \leq \|\boldsymbol{X}\|_F^2$, where $\|\boldsymbol{X}\|_F$ denotes the Frobenius norm of $\boldsymbol{X}$ defined by $\|\boldsymbol{X}\|_F^2 = \sum_{i=1}^n \sum_{j=1}^p x_{ij}^2$. The Frobenius norm and, hence, this estimate of $\|\boldsymbol{A}\|$ can be computed using only $\mathcal{O}(np)$ floating point operations.

In practice, we stop Algorithm 2 after a maximum number of iterations are performed or a sufficiently suboptimal solution is identified. Recall, that $\boldsymbol{\beta}^*$ minimizes $F(\boldsymbol{\beta})$ if $\mathbf{0} \in \partial F(\boldsymbol{\beta})$. On the other hand, we know that $\boldsymbol{A}\boldsymbol{\beta}^* + \boldsymbol{d} + \lambda\ \text{sign}(\boldsymbol{\beta}^*) \in \partial F(\boldsymbol{\beta}^*)$ by the structure of the subgradients of $\|\boldsymbol{\beta}\|_1$. This implies that we can terminate our proximal gradient update scheme if we find $\boldsymbol{\beta}^*$ such that $\boldsymbol{A}\boldsymbol{\beta}^* + \boldsymbol{d} + \lambda\ \text{sign}(\boldsymbol{\beta}^*)$ is close to $\mathbf{0}$. Specifically, we stop the iterative scheme after the $i$th iteration if

$$\|\boldsymbol{A}\boldsymbol{\beta}^i + \boldsymbol{d} + \lambda\ \text{sign}(\boldsymbol{\beta}^i)\|_\infty = \max_j |(\boldsymbol{A}\boldsymbol{\beta}^i + \boldsymbol{d} + \lambda\ \text{sign}(\boldsymbol{\beta}^i))_j| \leq p\epsilon$$

for given stopping tolerance $\epsilon > 0$.

### 3.2 The accelerated proximal gradient method

The similarity of our method to iterative soft thresholding and, more generally, our use of proximal gradient steps to mimic the gradient method for minimization of our nonsmooth objective suggests that we may be able to use momentum terms to accelerate convergence of our iterates. In particular, we modify the fast iterative soft thresholding algorithm (FISTA) described in Beck and Teboulle (2009), Section 4 to solve our subproblem. This approach extends a variety of accelerated gradient descent methods, most notably those of Nesterov (1983, 2005, 2013), to minimization of composite convex functions; for further details regarding the acceleration process and motivation for why such acceleration is possible, we direct the reader to the references (Allen-Zhu and Orecchia 2017; Bubeck et al. 2015; Flammarion and Bach 2015; Lessard et al. 2016; O'Donoghue and Candes 2015; Su et al. 2014; Tseng 2008).

We accelerate convergence of our iterates by taking a proximal gradient step from an extrapolation of the last two iterates. Applied to (7), the accelerated proximal gradient method features updates of the form

$$\boldsymbol{y}^{i+1} = \boldsymbol{x}^i + \omega_i(\boldsymbol{x}^i - \boldsymbol{x}^{i-1}) \tag{13}$$

$$\boldsymbol{x}^{i+1} = \underset{\alpha g}{\text{prox}}(\boldsymbol{y}^{i+1} - \alpha\nabla f(\boldsymbol{y}^{i+1})), \tag{14}$$

where $\omega_i \in [0, 1)$ is an extrapolation parameter, a standard choice of this parameter is $i/(i+3)$. Applying this modification to our original proximal gradient algorithm

**Algorithm 4:** Accelerated proximal gradient method for solving (6) with constant step size

**Data**: Given initial iterates $\boldsymbol{\beta}^0 = \boldsymbol{\beta}^1$, step length $\alpha$, sequence of extrapolation parameters $\{\omega_i\}_{i=0}^{\infty}$, stopping tolerance $\epsilon$, and maximum number of iterations $N$.
**Result**: Solution $\boldsymbol{\beta}^*$ of (6).
**for** $i = 1, 2 \ldots, N$
  Update momentum term by (13):

$$\boldsymbol{y}^{i+1} = \boldsymbol{\beta}^i + \omega_i(\boldsymbol{\beta}^i - \boldsymbol{\beta}^{t-1});$$

  Update gradient term by (11):

$$\boldsymbol{p}^{i+1} = \boldsymbol{y}^{i+1} - \alpha(\boldsymbol{A}\boldsymbol{y}^{i+1} + \boldsymbol{d});$$

  Update iterate using proximal gradient step (10):

$$\boldsymbol{\beta}^{i+1} = \text{sign}(\boldsymbol{p}^{i+1}) \max\{|\boldsymbol{p}^{i+1}| - \lambda\alpha\mathbf{1}, \mathbf{0}\};$$

  Terminate if current iterate is approximately stationary:
  **if** $\|\boldsymbol{A}\boldsymbol{\beta}^{i+1} + \boldsymbol{d} + \lambda\ \text{sign}(\boldsymbol{\beta}^{i+1})\|_\infty < p\epsilon$:
    The algorithm has converged;
    **break**;
  **end**
**end**

yields Algorithm 4. Modifying the backtracking line search of Algorithm 3 to use the accelerated proximal gradient update yields Algorithm 5. It can be shown that the sequence of iterates generated by either of these algorithms converges in value to the optimal solution of (6) at rate $\mathcal{O}(1/i^2)$.

**Theorem 4** (Specialization of Beck and Teboulle 2009, Theorem 4.4) *Let* $\{\boldsymbol{\beta}^i\}$ *be generated by Algorithm 4 and constant step size* $\alpha_i = \alpha \in (0, 1/\|\boldsymbol{A}\|)$ *or generated using backtracking line search by Algorithm 5 with initial iterate* $\boldsymbol{\beta}^0$. *Then there exists constant* $c > 0$ *such that*

$$F(\boldsymbol{\beta}^i) - F(\boldsymbol{\beta}^*) \leq \frac{c\|\boldsymbol{A}\|\|\boldsymbol{\beta}^0 - \boldsymbol{\beta}^*\|^2}{i^2} \tag{15}$$

*for any* $i \geq 1$ *and minimizer* $\boldsymbol{\beta}^*$ *of* $F$.

### 3.3 The alternating direction method of multipliers

We conclude by proposing a third algorithm for minimization of (9) based on the *alternating direction method of multipliers* (ADMM). The ADMM is designed to minimize separable objective functions under linear coupling constraints, i.e., problems of the form

$$\min_{\boldsymbol{x}\in\mathbf{R}^p, y\in\mathbf{R}^m} \Big\{ f(\boldsymbol{x}) + g(\boldsymbol{y}) : \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} = \boldsymbol{c} \Big\}, \tag{16}$$

**Algorithm 5:** Accelerated proximal gradient method for solving (6) with backtracking line search

**Data**: Initial iterates $\boldsymbol{\beta}^0 = \boldsymbol{\beta}^1$, initial Lipschitz constant $L_0 > 0$, scaling parameter $\eta > 1$, sequence of extrapolation parameters $\{\omega_i\}_{i=0}^{\infty}$, stopping tolerance $\epsilon$, and maximum number of iterations $N$.
**Result**: Solution $\boldsymbol{\beta}^*$ of (6).
**for** $i = 0, 1, 2 \ldots N$
  Update $\boldsymbol{\beta}^{i+1}$ using accelerated proximal gradient step and backtracking line search:
  **for** $k = 0, 1, 2 \ldots$ until step size accepted
    Update step length:

$$\bar{L} = \eta^k L_i, \qquad \alpha = \frac{1}{\bar{L}};$$

    Update iterate using accelerated proximal gradient step (13), (14):

$$\begin{aligned} \boldsymbol{y}^{i+1} &= \boldsymbol{\beta}^i + \omega_i(\boldsymbol{\beta}^i - \boldsymbol{\beta}^{t-1}) \\ \boldsymbol{p}^{i+1} &= \boldsymbol{y}^{i+1} - \alpha(\boldsymbol{A}\boldsymbol{y}^{i+1} + \boldsymbol{d}) \\ \boldsymbol{\beta}^{i+1} &= \operatorname{sign}(\boldsymbol{p}^{i+1}) \max\{|\boldsymbol{p}^{i+1}| - \lambda\alpha\mathbf{1}, \mathbf{0}\}; \end{aligned}$$

    Determine whether to accept update or increment step length:
    **if** $(\boldsymbol{\beta}^{i+1} - \boldsymbol{y}^{i+1})^T \left(\frac{\bar{L}}{2}\boldsymbol{I} - \boldsymbol{A}\right)(\boldsymbol{\beta}^{i+1} - \boldsymbol{y}^{i+1}) \geq 0$
      Accept update: set $L_{i+1} = \bar{L}$ and $\alpha_i = \bar{\alpha}$;
      **break**;
    **end**
  **end**
  Terminate if current iterate is approximately stationary:
  **if** $\|\boldsymbol{A}\boldsymbol{\beta}^{i+1} + \boldsymbol{d} + \lambda \operatorname{sign}(\boldsymbol{\beta}^{i+1})\|_\infty < p\epsilon$:
    The algorithm has converged;
    **break**;
  **end**
**end**

via an approximate dual gradient ascent, where $f : \mathbf{R}^p \to \mathbf{R}$, $g : \mathbf{R}^m \to \mathbf{R}$, $\boldsymbol{A} \in \mathbf{R}^{r\times p}$, $\boldsymbol{B} \in \mathbf{R}^{r\times m}$, and $\boldsymbol{c} \in \mathbf{R}^r$; we direct the reader to the survey Boyd et al. (2011) for more details regarding the ADMM.

The minimization of the composite function $F$ defined in (9) can be written as the unconstrained optimization problem

$$\min_{\boldsymbol{\beta}\in\mathbf{R}^p} F(\boldsymbol{\beta}) = \min_{\boldsymbol{\beta}\in\mathbf{R}^p} \frac{1}{2}\boldsymbol{\beta}^T\boldsymbol{A}\boldsymbol{\beta} + \boldsymbol{d}^T\boldsymbol{\beta} + \lambda\|\boldsymbol{\beta}\|_1. \tag{17}$$

We can rewrite (17) in an equivalent form appropriate for the ADMM by splitting the decision variable $\boldsymbol{\beta} \in \mathbf{R}^p$ as two new variables $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{R}^p$ with an accompanying linear coupling constraint $\boldsymbol{x} = \boldsymbol{y}$. Following this change of variables, we can express (17) as

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{y}\in\mathbf{R}^p} \ & \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} + \boldsymbol{d}^T\boldsymbol{x} + \lambda\|\boldsymbol{y}\|_1 \\ \text{s.t. } & \boldsymbol{x} - \boldsymbol{y} = \mathbf{0}. \end{aligned} \tag{18}$$

The ADMM generates a sequence of iterates using approximate dual gradient ascent as follows. The augmented Lagrangian of (18) is defined by

$$\mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{y}, z) = \frac{1}{2}\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} + \boldsymbol{d}^T \boldsymbol{x} + \lambda\|\boldsymbol{y}\|_1 + z^T(\boldsymbol{x} - \boldsymbol{y}) + \frac{\mu}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2$$

for all $\boldsymbol{x}, \boldsymbol{y}, z \in \mathbf{R}^p$; here, $\mu > 0$ is a penalty parameter controlling the emphasis on enforcing feasibility of the primal iterates $\boldsymbol{x}$ and $\boldsymbol{y}$. To approximate the gradient of the dual functional of (18), we alternately minimize the augmented Lagrangian with respect to $\boldsymbol{x}$ and $\boldsymbol{y}$. We then update the dual variable $z$ using a dual ascent step using this approximate gradient.

Suppose that we have the iterates $(\boldsymbol{x}^i, \boldsymbol{y}^i, z^i)$ after $i$ iterations. To update $\boldsymbol{x}$, we take

$$\boldsymbol{x}^{i+1} = \underset{\boldsymbol{x}\in\mathbf{R}^p}{\arg\min}\, \mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{y}^i, z^i) = \underset{\boldsymbol{x}\in\mathbf{R}^p}{\arg\min}\, \frac{1}{2}\boldsymbol{x}^T(\mu\boldsymbol{I} + \boldsymbol{A})\boldsymbol{x} - \boldsymbol{x}^T(-\boldsymbol{d} + \mu\boldsymbol{y}^i - z^i).$$

Applying the first order necessary and sufficient conditions for optimality, we see that $\boldsymbol{x}^{i+1}$ must satisfy

$$(\mu\boldsymbol{I} + \boldsymbol{A})\boldsymbol{x}^{i+1} = -\boldsymbol{d} + \mu\boldsymbol{y}^i - z^i. \tag{19}$$

Thus, $\boldsymbol{x}^{i+1}$ is obtained as the solution of a linear system. Note that the coefficient matrix $\mu\boldsymbol{I} + \boldsymbol{A}$ is independent of the iteration number $i$; we take the Cholesky decomposition of $\mu\boldsymbol{I} + \boldsymbol{A} = \boldsymbol{B}\boldsymbol{B}^T$ during a preprocessing step and obtain $\boldsymbol{x}^{i+1}$ by solving the two triangular systems given by

$$\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{x}^{i+1} = -\boldsymbol{d} + \mu\boldsymbol{y}^i - z^i.$$

When the generalized elastic net matrix $\boldsymbol{\Omega}$ is diagonal, or $\boldsymbol{M} := \mu\boldsymbol{I} + 2\gamma\boldsymbol{\Omega}$ is otherwise easy to invert, we can invoke the Sherman-Morrison-Woodbury formula (see Golub and Van Loan 2013, Section 2.1.4) to solve this linear system more efficiently; more details will be provided in Sect. 3.4. In particular, we see that

$$(\mu\boldsymbol{I} + 2\gamma\boldsymbol{\Omega} + 2\boldsymbol{X}^T\boldsymbol{X})^{-1} = \boldsymbol{M}^{-1} - 2\boldsymbol{M}^{-1}\boldsymbol{X}^T(\boldsymbol{I} + 2\boldsymbol{X}\boldsymbol{M}^{-1}\boldsymbol{X}^T)^{-1}\boldsymbol{X}\boldsymbol{M}^{-1};$$

computing this inverse only requires computing the inverse of $\boldsymbol{M}$ and the inverse of the $n \times n$ matrix $\boldsymbol{I} + 2\boldsymbol{X}\boldsymbol{M}^{-1}\boldsymbol{X}^T$.

Next $\boldsymbol{y}$ is updated by

$$\boldsymbol{y}^{i+1} = \underset{y\in\mathbf{R}^p}{\arg\min}\, \mathcal{L}_\mu(\boldsymbol{x}^{i+1}, \boldsymbol{y}, z^i) = \underset{\boldsymbol{y}}{\arg\min}\, \lambda\|\boldsymbol{y}\|_1 + \frac{\mu}{2}\|\boldsymbol{y} - \boldsymbol{x}^{i+1} - z^i/\mu\|^2.$$

That is, $\boldsymbol{y}^{i+1}$ is updated as the value of the soft thresholding operator of the $\ell_1$-norm at $z^i/\mu + \boldsymbol{x}^{i+1}$:

$$\boldsymbol{y}^{i+1} = S_{\lambda/\mu}(\boldsymbol{x}^{i+1} + z^i/\mu). \tag{20}$$

Finally, the dual variable $z$ is updated using the approximate dual ascent step

$$z^{i+1} = z^i + \mu(x^{i+1} - y^{i+1}). \tag{21}$$

Following each iteration, we check that the Karush-Kuhn-Tucker conditions for (16) have been approximately satisfied as a stopping criterion. Specifically, we check if the updated iterates $(x^{i+1}, y^{i+1}, z^{i+1})$ have satisfied primal and dual feasibility within relative tolerance of $\epsilon$ by checking if the inequalities

$$\begin{aligned} \|x^{i+1} - y^{i+1}\| &\leq \epsilon \ \max\{\|x^{i+1}\|, \|y^{i+1}\|\} \\ \mu\|y^{i+1} - y^i\| &\leq \epsilon \ \|y^{i+1}\|, \end{aligned}$$

respectively, are satisfied. This approach is summarized in Algorithm 6.

It is well-known that the ADMM generates a sequence of iterates which converge linearly to an optimal solution of (16) under certain strong convexity assumptions on $f$ and $g$ and rank assumptions on $\boldsymbol{A}$ and $\boldsymbol{B}$, all of which are satisfied by our problem (18) when $\boldsymbol{\Omega}$ is positive definite (see, for example, Deng and Yin (2012), Goldfarb et al. (2013), Nishihara et al. (2015), He and Yuan (2012)). It follows that the sequence of iterates $\{x^i, y^i, z^i\}$ generated by Algorithm 6 converges to a minimizer of $F(\boldsymbol{\beta})$; that is, $x^i - y^i \to \mathbf{0}$ and $F(x^i)$, $F(y^i)$ converge linearly to the minimum value of $F$. The following theorem gives a worst-case convergence rate for Algorithm 6.

**Theorem 5** (Specialization of He and Yuan 2012, Theorem 4.1) *Suppose* $(x^i, y^i, z^i)$ *is generated by Algorithm 6. Suppose further that the objective function of* (16), $F(x, y) = \frac{1}{2}x^T \boldsymbol{A}x + \boldsymbol{d}^T x + \lambda\|y\|_1$, *is strongly convex. Then the sequence of iterates satisfies* $x^i - y^i \to \mathbf{0}$ *and*

$$F(x^i, y^i) - F(x^*, y^*) \leq \frac{C\|\boldsymbol{A}\|\|x^0 - x^*\|^2}{i} \tag{22}$$

*for some constant* $C > 0$, *where* $(x^*, y^*)$ *is a minimizer of* (16).

If $F$ is not strongly convex, then we should expect Algorithm 6 to generate a sequence of iterates that converges sublinearly in value to the optimal value of (6).

### 3.4 Computational requirements

To motivate the use of our proposed proximal methods for the minimization of (6), we briefly sketch the computational costs of each of our methods. We will see that for certain choices of regularization parameters, the number of floating point operations needed scales linearly with the size of the data.

We begin by with the computational costs of the proximal gradient method (Algorithms 2 and 3). The computational requirements for the accelerated proximal gradient (Algorithms 4 and 5) and alternating direction method of multipliers (Algorithm 6) can be calculated in a similar fashion; a full calculation of the complexity of each iteration of each method can be found in "Appendix A". The most expensive operation of each

**Algorithm 6:** Alternating direction method of multipliers for solving (18)

**Data**: Given initial iterates $\boldsymbol{x}^0 = \boldsymbol{y}^0$, step length $\mu$, stopping tolerance $\epsilon$, and maximum number of iterations $N$.
**Result**: Solution $\boldsymbol{\beta}^* = \boldsymbol{x}^* = \boldsymbol{y}^*$ of (6).
**for** $i = 0, 1, 2 \ldots, N$
  Update $\boldsymbol{x}$ by (19):

$$(\mu \boldsymbol{I} + \boldsymbol{A})\boldsymbol{x}^{i+1} = -\boldsymbol{d} + \mu \boldsymbol{y}^i - \boldsymbol{z}^i;$$

  Update $\boldsymbol{y}$ using soft thresholding (20):

$$\boldsymbol{y}^{i+1} = S_{\lambda/\mu}(\boldsymbol{x}^{i+1} + \boldsymbol{z}^i/\mu);$$

  Update $\boldsymbol{z}$ using approximate dual ascent (21):

$$\boldsymbol{z}^{i+1} = \boldsymbol{z}^i + \mu(\boldsymbol{x}^{i+1} - \boldsymbol{y}^{i+1});$$

  Test stopping criterion:
  **if** $\|\boldsymbol{x}^{i+1} - \boldsymbol{y}^{i+1}\| \leq \epsilon \max\{\|\boldsymbol{x}^{i+1}\|, \|\boldsymbol{y}^{i+1}\|\}$ *and* $\mu\|\boldsymbol{y}^{i+1} - \boldsymbol{y}^i\| \leq \epsilon\|\boldsymbol{y}^{i+1}\|$
    The algorithm has converged;
    **break**;
  **end**
**end**

**Table 1** Upper bounds on total number of floating point operations required to calculate an $\epsilon$-suboptimal solution (PG, APG, ADMM) or solution containing $m$ nonzero entries (LARS) using of (6) diagonal regularization matrix $\boldsymbol{\Omega}$ and dense, unstructured $\boldsymbol{\Omega}$

| Method | PG | APG | ADMM | LARS |
|---|---|---|---|---|
| Diagonal $\boldsymbol{\Omega}$ | $\mathcal{O}(np/\epsilon)$ | $\mathcal{O}(np/\sqrt{\epsilon})$ | $\mathcal{O}(n^2 p/\epsilon)$ | $\mathcal{O}(mnp + m^3)$ |
| Dense $\boldsymbol{\Omega}$ | $\mathcal{O}(p^2/\epsilon)$ | $\mathcal{O}(p^2/\sqrt{\epsilon})$ | $\mathcal{O}(p^3 + p^2/\epsilon)$ | $\mathcal{O}(mnp + m^3)$ |

iteration of Algorithm 2 is the calculation of the gradient: $\nabla f(\boldsymbol{\beta}) = \boldsymbol{A}\boldsymbol{\beta}$. This requires $\mathcal{O}(np)$ floating point operations (flops) if the regularization matrix $\boldsymbol{\Omega}$ is a diagonal matrix (and $\mathcal{O}(p^2)$ flops for unstructured $\boldsymbol{\Omega}$). On the other hand, Theorem 3 implies that Algorithms 2 and 3 generate an $\epsilon$-suboptimal solution of (6) within $\mathcal{O}(1/\epsilon)$ iterations. Putting everything together, we see that Algorithms 2 and 3 yield $\epsilon$-suboptimal solutions using at most $\mathcal{O}(np/\epsilon)$ flops if $\boldsymbol{\Omega}$ is a diagonal matrix. Performing similar calculations for Algorithms 4, 5, and 6 yields the total complexity estimates for each method, found in Table 1. We remind the reader that the flop counts given in Table 1 are restricted to the case when $\boldsymbol{\Omega}$ is diagonal or easy to invert; all methods scale cubically in $p$ if $\boldsymbol{\Omega}$ is unstructured and/or difficult to invert.

The complexity estimates found in Table 1 establish that the accelerated proximal gradient method is more efficient for solution of (6) than least angle regression if

$$\frac{np}{\sqrt{\epsilon}} << mnp + m^3, \tag{23}$$

where $m$ is the number of nonzero entries of the optimal solution $\boldsymbol{\beta}^*$ when $\boldsymbol{\Omega}$ is a diagonal matrix. In particular, if $m$ is moderately large, e.g., $m = c(np)^{1/3}$ for sufficiently large $c$, APG is significantly more efficient for solution of (6) than LARS. In practice, this improvement in computational complexity is large when $p$ is large (e.g., $p > 1000$).

In the case that $n$ is large (or both $n$ and $p$ are large), Table 1 suggests that ADMM should be prohibitively expensive, relative to the other methods considered. We should note that our implementation of the ADMM for solving (6) is optimized for the case when $n$ is much smaller than $p$. In particular, our use of the Sherman-Morrison-Woodbury formula to update $\boldsymbol{x}$ in Algorithm 6 explicitly relies on the assumption that $n < p$.

Section 4 provides a detailed empirical analysis of the computational complexity of these algorithms for solving (6).

## 4 Numerical analysis

We next compare the performance of our proposed approaches with standard methods for penalized discriminant analysis in several numerical experiments. In particular, we compare the implementations of the block coordinate descent method Algorithm 1, where each discriminant direction $\boldsymbol{\beta}$ is updated using either the proximal gradient method with constant step size, Algorithm 2 (PG), the proximal gradient method with backtracking line search, Algorithm 3 (PGB), the accelerated proximal method with constant step size, Algorithm 4 (APG), the accelerated proximal method with backtracking, Algorithm 5 (APGB), and the alternating direction method of multipliers, Algorithm 6, (ADMM), with the least angle regression based algorithm (LARS) for solving the sparse optimal scoring problem proposed in Clemmensen et al. (2011). We choose LARS as our baseline for comparison due to its popular use as a heuristic for the sparse optimal scoring problem and due to the ease of its application in our numerical trials using the `sparseLDA` package Clemmensen (2008) in Matlab and R.

### 4.1 Gaussian data

We first perform simulations investigating efficacy of our heuristics for classification of Gaussian data. In each experiment, we generate data consisting of $p$-dimensional vectors from one of $K$ multivariate normal distributions. Specifically, we obtain training observations corresponding to the $i$th class, $i = 1, 2, \ldots, K$, by sampling 25 observations from the multivariate normal distribution with mean $\boldsymbol{\mu_i} \in \mathbf{R}^p$ satisfying

$$[\boldsymbol{\mu_i}]_j = \begin{cases} 0.7, & \text{if } 100(i-1) < j \leq 100i \\ 0, & \text{otherwise,} \end{cases} \tag{24}$$

for all $j = 1, 2, \ldots, p$, and covariance matrix $\boldsymbol{\Sigma} \in \mathbf{R}^{p \times p}$ chosen so that all features are correlated with $\Sigma_{ij} = r$ for all $i \neq j$ and $\Sigma_{ii} = 1$ for all $i$. We conduct the

experiment for all $K \in \{2, 4\}$, $r \in \{0, 0.1, 0.5, 0.9\}$. For each experiment, we sample 250 testing observations from each class in the same manner as the training data. We set $p = 1500$ in each simulation. For each $(K, r)$ pair, we generate 20 data sets and use nearest centroid classification following projection onto the span of the discriminant directions calculated using Algorithm 1 and PG, PGB, APG, APGB, ADMM, or LARS to solve (6), or SZVD.

The sparse discriminant analysis heuristics require training of the regularization parameters $\gamma$, $\boldsymbol{\Omega}$, and $\lambda$. In all experiments, we set $\gamma = 10^{-3}$ and $\boldsymbol{\Omega}$ to be the $p \times p$ identity matrix $\boldsymbol{\Omega} = \boldsymbol{I}$. We train the remaining parameter $\lambda$ using 5 fold cross validation. Specifically, we choose $\lambda$ from a set of potential $\lambda$ of the form $\bar{\lambda}/2^c$ for $c = 3, 2, \ldots, 0, -1$, and $\bar{\lambda}$ chosen so that the problem has nontrivial solution for all considered $\lambda$. Note that (9) has optimal solution given by $\boldsymbol{\beta}^* = \boldsymbol{A}^{-1}\boldsymbol{d}$ if we set $\lambda = 0$; this implies that choosing

$$\bar{\lambda} = \frac{(\boldsymbol{\beta}^*)^T \boldsymbol{d} - \frac{1}{2}(\boldsymbol{\beta}^*)^T \boldsymbol{A}\boldsymbol{\beta}^*}{\|\boldsymbol{\beta}^*\|_1} \tag{25}$$

ensures that there exists at least one solution $\boldsymbol{\beta}^*$ with value strictly less than zero. We choose the value of $\lambda$ with fewest average number of misclassification errors over training-validation splits amongst all $\lambda$ which yield discriminant vectors containing at most 25% nonzero entries; in the event of a tie, we select the value of $\lambda$ which yields discriminant vectors with smallest average cardinality among all with minimum validation score. Note that we could have applied other resampling methods, e.g., boot strapping, to train the regularization parameter using the same criteria with minimal changes to the experiment. To reduce computation time, we use less conservative stopping criteria during the cross validation procedure. We terminate each proximal algorithm in the inner loop after 1000 iterations or a $10^{-4}$ suboptimal solution is obtained. We perform exactly one iteration of the outer block coordinate descent loop if $K = 2$ and the outer loop is stopped after a maximum number of 250 iterations or a $10^{-3}$ suboptimal solution has been found if $K = 4$. After the optimal value of $\lambda$ is determined using cross validation, we train using the full training set; we terminate the algorithm after 1000 iterations or a $10^{-5}$ suboptimal solution is obtained. The augmented Lagrangian parameter $\mu = 2$ was used for the ADMM method in all experiments. We use the value of $\bar{L} = 0.25$ for the initial estimate of the Lipschitz constant and $\eta = 1.25$ for the scalar multiplier in the backtracking line search.

The LARS algorithm for updating $\boldsymbol{\beta}$ was terminated after the convergence criterion is met with tolerance $10^{-3}$ or an iterate with more than $0.25p$ nonzero entries was found. As with the other methods, we perform either one outer iteration ($K = 2$) or at most 250 outer iterations, stopping when a $10^{-3}$ suboptimal solution is found ($K = 4$). Since LARS generates the full regularization path for each value of $\boldsymbol{\theta}$, we do not need to perform cross validation to tune the parameter $\lambda$. Instead, we choose the value of $\boldsymbol{\beta}$ with minimum classification error on the *training* set as our optimal discriminant value at termination.

These stopping criteria and parameter choices were chosen empirically in order to yield accurate classifiers using a minimal number of iterations; in particular, using modest stopping tolerances limits the number of iterations performed, which tends to

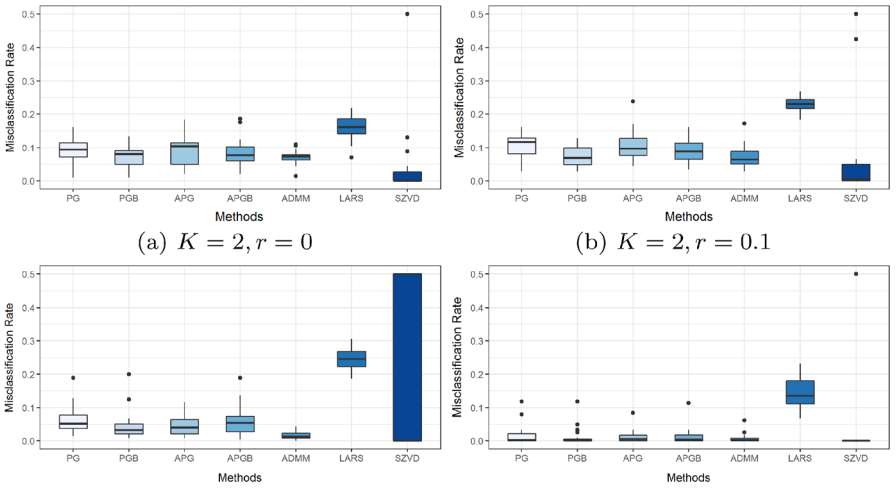
limit overfitting in practice. Indeed, we should carefully note that the sparse optimal scoring problem acts as a *proxy* for the classification task, but does not explicitly optimize classification accuracy. We often observe better classification rates in practice using modest stopping tolerances than we do when solving (1) to high precision; solutions obtained using modest stopping criteria also tend to generalize better to out-of-sample classification. We should also note that the set of (approximate) solutions $(\boldsymbol{\theta}_i^*, \boldsymbol{\beta}_i^*)$, $i = 1, 2, \ldots, K-1$ generated by each heuristic is feasible for (2) and (1) regardless of heuristic used (up to small rounding errors), since we maintain a sequence of feasible iterates at each step of Algorithm 1.

We also include the Sparse Zero-Variance Discriminant Analysis (SZVD) method proposed in Ames and Hong (2016) in our comparisons. We train the regularization parameter $\gamma$ in SZVD in a fashion similar to that above. We set the maximum value of the regularization parameter $\gamma$ to

$$\bar{\gamma} = \frac{\hat{\boldsymbol{\beta}}^T \boldsymbol{B} \hat{\boldsymbol{\beta}}}{\|\hat{\boldsymbol{\beta}}\|_1}, \tag{26}$$

where $\hat{\boldsymbol{\beta}}$ is the optimal solution of the unpenalized SZVD problem and $\boldsymbol{B}$ is the sample between-class covariance matrix of the training data. We choose $\gamma$ from the exponentially spaced grid $\bar{\gamma}/2^c$ for $c = 3, 2, 1, 0, -1$ using 5 fold cross-validation; this approach is consistent with that in Ames and Hong (2016). We select the value of $\gamma$ which minimizes misclassification error amongst all sets of discriminant vectors with at most 35% nonzero entries; this acceptable sparsity threshold is chosen to be higher than that in the SOS experiments, due to the tendency of SZVD to misconverge to the trivial all-zero solution for large values of $\gamma$. We stop SZVD after a maximum of 1000 iterations or a solution satisfying the stopping tolerance of $10^{-5}$ is obtained. We use the augmented Lagrangian penalty parameter $\beta = 1.25$ in SZVD in all experiments. All simulations and experiments in the following sections were performed using `Matlab 2019b` on a standard node of the high performance computing system at the Alabama Supercomputer Center. The `Matlab` implementation of Algorithm 1 can be obtained from https://github.com/gumeo/accSDA_matlab. We use the Matlab package `sparseLDA` Clemmensen (2008) with modification to return the full optimization path to solve (6) using the LARS algorithm.

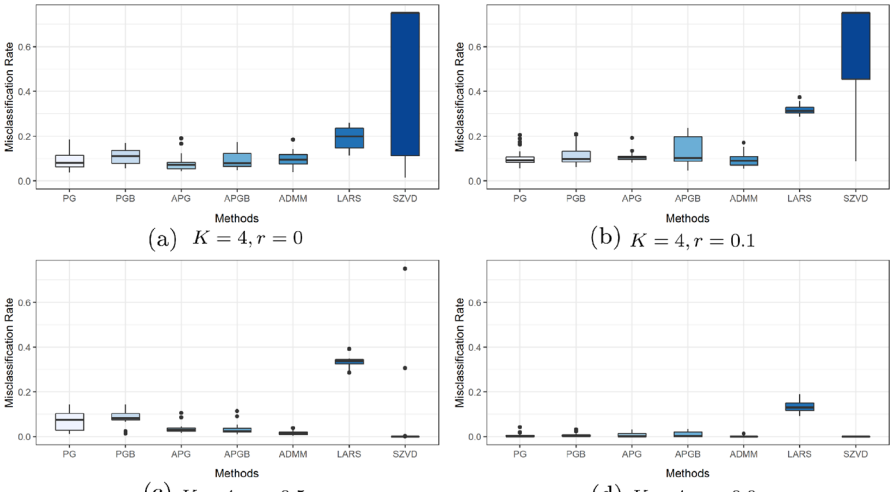
Figures 1, 2, 3, 4, 5, and 6 summarize the results of these experiments; complete tables of numerical results can be found in the electronic supplemental materials Online Resource 1 and Online Resource 2. The run-times in Figs. 5 and 6 are reported in seconds ($s$) and reflect the total computation time required for each method to perform cross validation to tune regularization parameters and train optimal scoring and discriminant vector pairs using the optimized regularization parameters. The number of nonzero features reported for all methods except SZVD was the count of discriminant vector elements with value not identical to 0. SZVD does not use the same soft thresholding step as the other methods and returns approximately sparse solutions with entries close to zero, but not exactly zero; we round entries with magnitude at most $10^{-5}$ when reporting cardinality of discriminant vectors obtained using SZVD.




(a) $K = 2, r = 0$ (b) $K = 2, r = 0.1$

(c) $K = 2, r = 0.5$ (d) $K = 2, r = 0.9$

**Fig. 1** Box plot of out-of-sample misclassification rate for 2-class Gaussian data with class means defined by (24) and covariance vector $\boldsymbol{\Sigma}$ for given values of $r$. For each data set, we use nearest centroid classification following projection onto discriminant vectors given by the sparse zero variance method (SZVD) or optimal scoring vectors calculated using the proximal gradient method with constant stepsize (PG), with backtracking line search (PGB), accelerated proximal method with constant stepsize (APG) and backtracking line search (APGB), alternating direction method of multipliers (ADMM), and least angle regression (LARS). In all experiments, $n_{train} = 50$ and $n_{test} = 500$




(a) $K = 4, r = 0$ (b) $K = 4, r = 0.1$

(c) $K = 4, r = 0.5$ (d) $K = 4, r = 0.9$

**Fig. 2** Box plot of out-of-sample misclassification rate using discriminant vectors calculated using APG, APGB, PG, PGB, ADMM, SZVD, and LARS for 4-class Gaussian data with class means defined by (24) and covariance vector $\boldsymbol{\Sigma}$ for given values of $r$. In all experiments, $n_{train} = 100$ and $n_{test} = 1000$

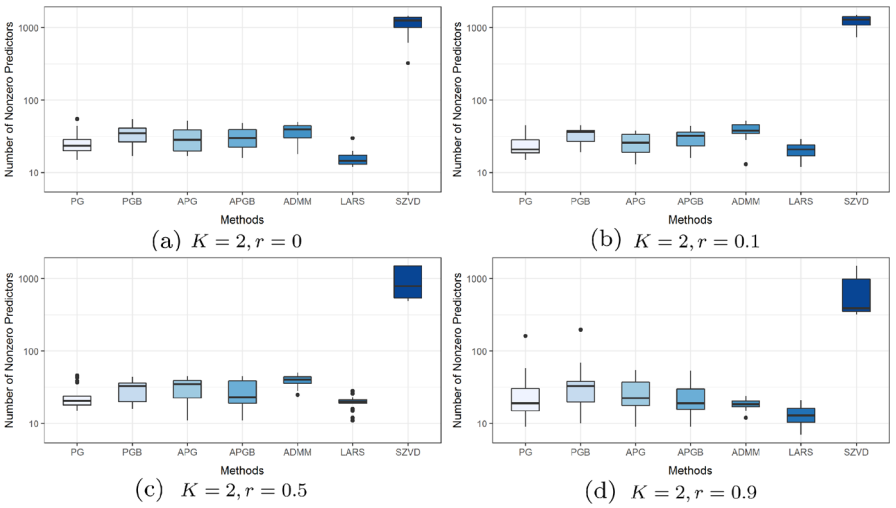



**Fig. 3** Box plot of number of nonzero entries of discriminant vectors and optimal scoring vectors calculated using APG, APGB, PG, PGB, ADMM, SZVD, and LARS for classifying 2-class Gaussian data with mean-vectors defined by (24) and covariance vector $\boldsymbol{\Sigma}$ for given values of $r$

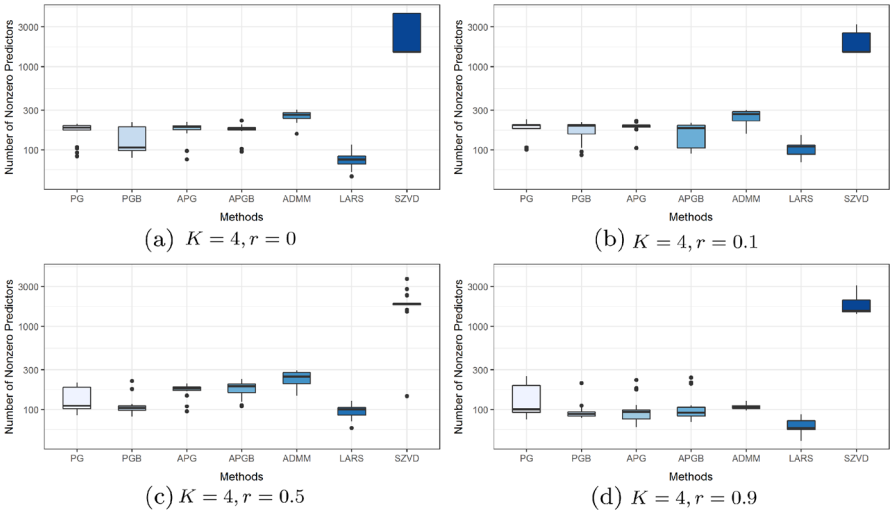
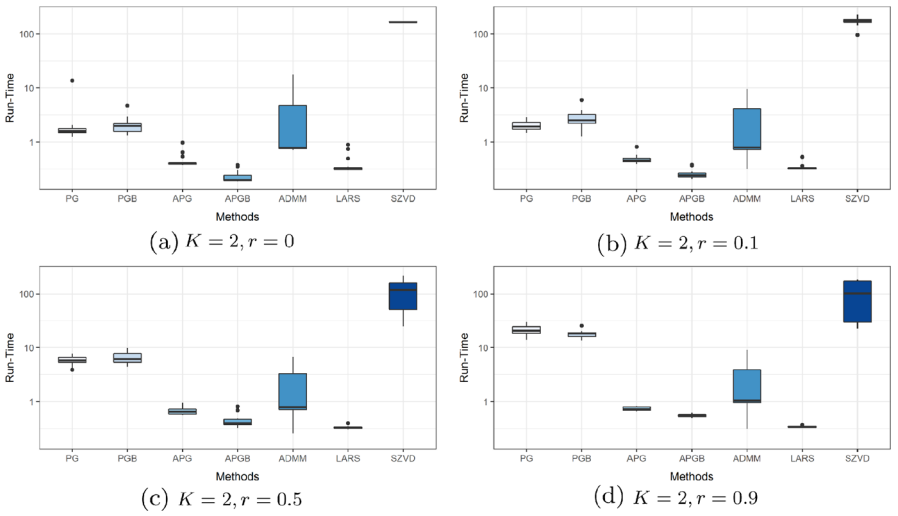



**Fig. 4** Box plot of number of nonzero entries of discriminant vectors and optimal scoring vectors calculated using APG, APGB, PG, PGB, ADMM, SZVD, and LARS for classifying 4-class Gaussian data with mean-vectors defined by (24) and covariance vector $\boldsymbol{\Sigma}$ for given values of $r$

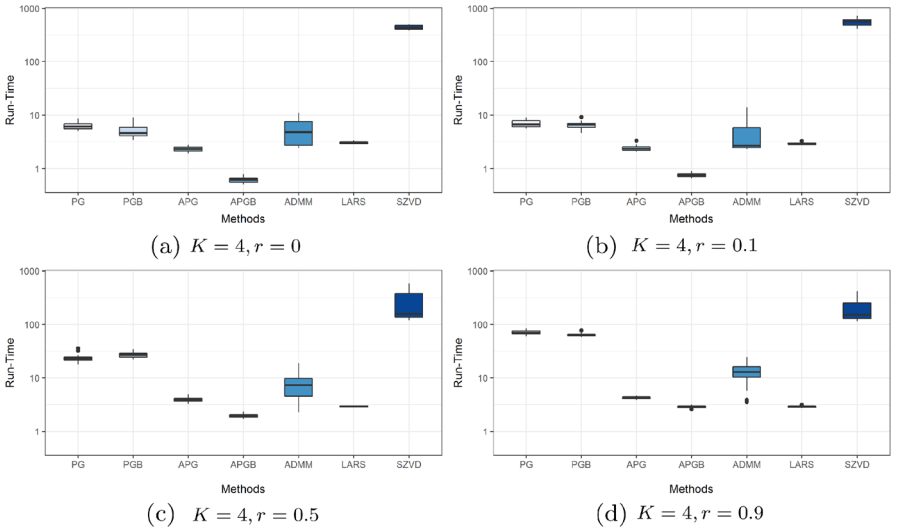
From these simulations, we see that the the classical solution of the SOS problem using LARS tends to yield fewer nonzero predictor variables than the APG and ADMM proximal methods (see Fig. 3 and 4), while requiring comparable computation (see Fig. 5 and 6); we note that the average cardinality and run-times observed are typically within one standard deviation of each other (as indicated by the error bars in the plots). On the other hand, the classifiers provided the LARS heuristic tend to have significantly




(a) $K = 2, r = 0$ (b) $K = 2, r = 0.1$

(c) $K = 2, r = 0.5$ (d) $K = 2, r = 0.9$

**Fig. 5** Box plots of run-time in seconds (s) for calculation of discriminant vectors and optimal scoring vectors (with error bars of length one standard deviation) using APG, APGB, PG, PGB, ADMM, SZVD, and LARS for classifying 2-class Gaussian data with mean-vectors defined by (24) and covariance vector $\boldsymbol{\Sigma}$ for given values of $r$




(a) $K = 4, r = 0$ (b) $K = 4, r = 0.1$

(c) $K = 4, r = 0.5$ (d) $K = 4, r = 0.9$

**Fig. 6** Box plots of run-time in seconds (s) for calculation of discriminant vectors and optimal scoring vectors (with error bars of length one standard deviation) using APG, APGB, PG, PGB, ADMM, SZVD, and LARS for classifying 4-class Gaussian data with mean-vectors defined by (24) and covariance vector $\boldsymbol{\Sigma}$ for given values of $r$

higher misclassification rate than the proposed proximal methods (see Fig. 1 and 2). We suspect that this is due to our method for choosing the regularization parameter $\lambda$. Here, we choose the value of $\lambda$ that minimizes in-sample classification accuracy, and break any ties by choosing the value of $\lambda$ which yields the sparsest classifier. This appears to cause some overfitting of the classifier to the training data, where the in-sample high accuracy does not generalize to high out-of-sample accuracy. In an earlier draft of this manuscript, we considered training $\lambda$ via cross-validation in a manner similar to that used for PG, PGB, APG, APGB, and ADMM; this produced a far lower classification rate, comparable to the other methods, although at a significantly higher computational cost, as were required to calculate the full regularization path for each fold and each choice of $\lambda$. In another previous draft, we chose $\lambda$ in our LARS-based analysis to minimize out-of-sample error. Again, this leads to a significant decrease in misclassification rate, although it is unfair to compare this approach to other methods which do not use this information to choose $\lambda$, and may not be used in practical applications where an out-of-sample ground truth is unavailable.

In all experiments, the sparse zero-variance discriminant analysis (SZVD) heuristic performed poorly in terms of computation, sparsity, and accuracy. The observed increased run-times agree with that predicted in Table 1. On the other hand, the relatively high density and misclassification rate can be explained by the tendency of SZVD to misconverge to an all-zeros solution when $\lambda$ is large. We observe a moderate number of trials where SZVD yields an all-zero solution (with high error rate), and remaining trials generating discriminant vectors with many nonzero entries (corresponding to relatively small $\lambda$).

We further investigate this phenomena in the following sections.

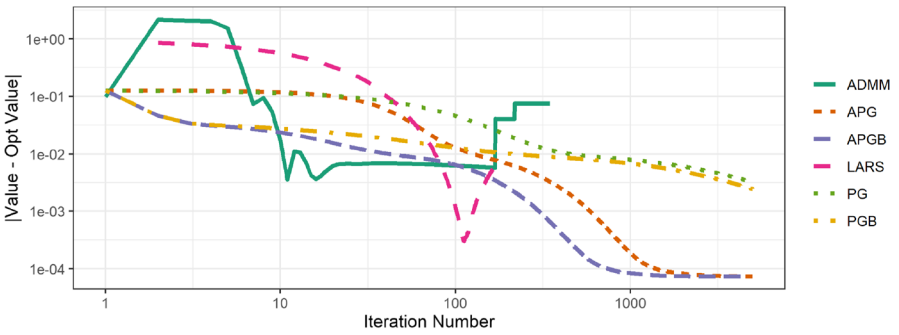
### 4.2 Convergence experiments

The empirical results of Sect. 4.1 suggest that the use of our proposed proximal methods (PG, PGB, APG, APGB, ADMM) for solution of subproblem (6) can lead to improvement in terms of classification accuracy and overall run-time over the least angle regression algorithm in certain settings. To further illustrate this improvement, we performed a series of experiments investigating the behaviour of the objective function of (2) during each iteration of these methods.

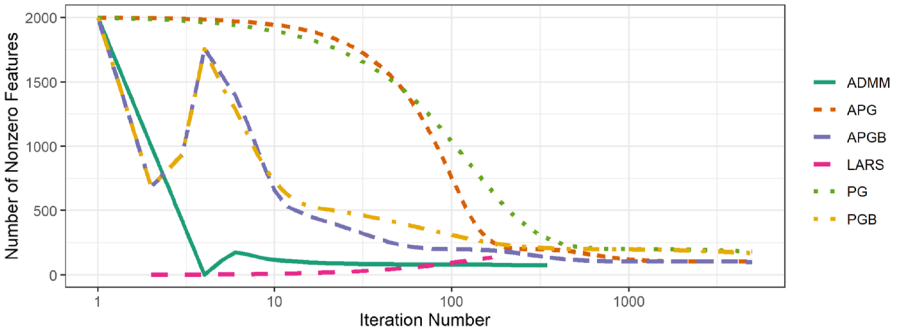
We generated random data sets with observations sampled from one of two normal distributions, $N(\boldsymbol{\mu_1}, \boldsymbol{\Sigma})$ or $N(\boldsymbol{\mu_2}, \boldsymbol{\Sigma})$. Specifically, we sampled $n = 200$ training observations from each of the $p$-dimensional multivariate Gaussian distributions for $p = 2000$ with mean vectors $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2} \in \mathbf{R}^p$, respectively, satisfying

$$[\boldsymbol{\mu_i}]_j = \begin{cases} 0.7, & \text{if } \lceil p/3 \rceil (i-1) < j \leq \lceil p/3 \rceil i \\ 0, & \text{otherwise,} \end{cases} \tag{27}$$

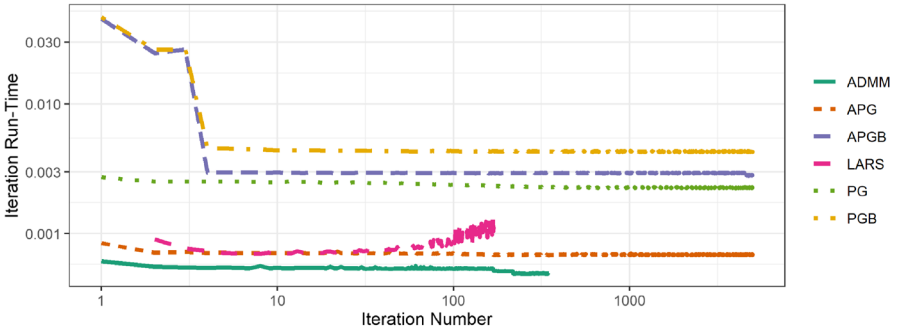
for all $j = 1, 2, \ldots, p$, and covariance matrix $\boldsymbol{\Sigma} \in \mathbf{R}^{p \times p}$ constructed as Section 4.1 with $r = 0.75$. For each data set, we train nearest centroid classifiers using discriminant vectors obtained by approximately solving (2) with each method used above (PG, PGB, APG, APGB, ADMM, LARS) to solve (6). We stop each method after




(a) Difference of objective value and optimal value




(b) Number of Nonzero Features




(c) Iteration Run-time

**Fig. 7** Absolute value of optimality gap, cardinality of iterate, and iteration run-time averaged over 50 calls to solve (6) for 25 different Gaussian data sets. We note that LARS terminates in the fewest number of iterations, followed by ADMM

either 10000 iterations have been performed or the stopping condition is met with tolerance $10^{-8}$. We use regularization parameters $\gamma = 10^{-3}$, $\boldsymbol{\Omega} = \boldsymbol{I}$ and we set $\lambda = 0.25\bar{\lambda}$, where $\bar{\lambda}$ is given by (25); we stop LARS when a solution with cardinality $0.25p = 500$ is found. We use augmented Lagrangian parameter $\mu = 2$ in ADMM. We use the parameters $\bar{L} = 0.25$ and $\eta = 1.25$ in the back tracking line search. Note that these stopping conditions are more strict than those used in the previous section

(Sect. 4.1); we use these stopping criteria to ensure that a large number of iterations are performed in order to obtain a clearer picture of the convergence properties of the various algorithms. We generate 25 problem instances and solve each problem 50 times using each method to control for natural variation in computation time and problem instances; each algorithm will generate the same sequence of iterates and solution each time called for each problem (up to sign changes due to random initialization of $\boldsymbol{\theta}$). We validate performance of our classifiers using balanced sets of 200 testing observations sampled from $N(\boldsymbol{\mu_1}, \boldsymbol{\Sigma})$ or $N(\boldsymbol{\mu_2}, \boldsymbol{\Sigma})$.

We chose these data sets to isolate the relationship between the performance of our proposed algorithms for solving (6) and the overall performance of the proposed block coordinate descent method and nearest centroid classification. Recall that, in the $K = 2$ class case, we calculate exactly one discriminant and scoring vector pair $(\boldsymbol{\beta}, \boldsymbol{\theta})$ before Algorithm 1 converges. By restricting our focus to a case where we solve exactly one instance of subproblem (6) using each method during each trial, we can directly compare the behavior of algorithms for solving (6).

We recorded the objective value of (6) and the cardinality of the current iterate $\boldsymbol{\beta}^i$ following the $i$th iteration of each algorithm for each method and data set, as well as the run-time of the $i$th iteration. We recorded the value of the augmented Lagrangian function for the ADMM, instead of the objective function value, to indicate the trade-off between optimizing the objective and forcing feasibility ($\boldsymbol{x} = \boldsymbol{y} = \boldsymbol{\beta}$) of the split decision variables $\boldsymbol{x}$ and $\boldsymbol{y}$. The results of these experiments are summarized in Fig. 7 and Table 2.

It is apparent from the results of these simulations that iterations of LARS are more expensive than those of the proximal methods APG and ADMM, especially when the cardinality of the iterate is relatively large. The per-iteration cost of LARS tends to increase since LARS is an active set method and gradually adds elements to the active set each iteration; the computational complexity is an increasing function of iteration number due to this corresponding increase in cardinality each iteration. We should also note that LARS tended to terminate more quickly (in terms of number of iterations) than the other methods, but that this is largely a consequence of the more conservative stopping criteria for the proximal methods. In particular, APG, APGB, ADMM all generate iterates with smaller optimality gap than the suboptimality at termination of LARS iterates, within fewer iterations. On the other hand, the per-iteration cost of each proximal method is largely consistent across iterations. The ADMM tended to terminate in fewer iterations and yield sparser solutions than the other proximal methods; the per-iteration cost of the ADMM is also less than (or comparable to) the other proximal methods in all trials. The cardinality of iterates generated by ADMM also decreased much more quickly than those generated by the other proximal methods; this may be due to the fact that the soft thresholding operator is applied directly to the iterate $\boldsymbol{y}^i$, rather than following a gradient step applied to the previous iterate or a weighted average of the previous two iterates as in the proximal gradient and accelerated gradient methods. The trajectory of suboptimality of ADMM iterates is not a smooth descent like the other proximal methods since we plot the absolute value of the gap between the augmented Lagrangian value and the average optimal value. Here, the augmented Lagrangian value of the ADMM iterates tend to initially increase followed by a several iterates of sharp decrease (with value typically less

**Table 2** Average run-time, cardinality of solution, and number of iterations before termination averaged over 25 Gaussian data sets, solved 50 times using each method.

| Method | Run-times (s) | Cardinality | Iterations |
|---|---|---|---|
| PG | 43.38 | 151.92 | 10000 |
| PGB | 63.98 | 141.52 | 1000 |
| APG | 28.79 | 100.88 | 10000 |
| APGB | 33.87 | 100.92 | 6633.92 |
| ADMM | 1.43 | 79.6 | 484.4 |
| LARS | 0.21 | 145.88 | 195.72 |

No methods returned classifiers yielding out-of-sample classification error in any trial. PG, PGB, APG failed to return a $10^{-8}$ suboptimal solution within 10000 iterations, while ADMM and LARS both converge within 500 iterations on average

than the optimal value), followed by gradual increase to the optimal value; plotting the absolute value of the optimality gap allows us plot using logarithmic scale. Similarly, the value of LARS iterates tends to decrease, and then increase prior to termination. This is a consequence of the stopping criteria, i.e., terminating when a sufficiently dense solution is found; here, we use the iterate at termination rather than the iterate with minimum value among LARS iterates as our discriminant vector when calculating misclassification rates and cardinality in Table 2.

These trials also suggest some modest value in the use of back tracking line searches. In each set of trials, the proximal gradient methods with back tracking line search terminated in fewer iterations than with a constant step size. However, the additional cost of performing the line search frequently caused the overall computational time of the back tracking methods to exceed that of the constant step size methods. This additional cost observed here is more dramatic than that observed in Sect. 4.1. We remind the reader that the reported computation time for the experiments of Sect. 4.1 includes all computation to perform cross validation to train the regularization parameter $\lambda$; the discrepancy between the timing results in Sect. 4.1 and here highlights a potential sensitivity of Algorithm 1 to choice of $\lambda$, and variation in training data (in this case with respect to training and validation splits in the cross validation scheme).

### 4.3 Differences between discriminant vectors due to algorithm choice

At this point, we should note that Subproblem (6) is strongly convex by the choice of regularization parameter $\boldsymbol{\Omega} = \boldsymbol{I}$ in all experiments considered so far. As a consequence, (6) has a *unique* solution. One would naively expect Algorithm 1 to generate the same discriminant vector regardless of choice of algorithm for solving (6) if all other input parameters are chosen consistently. However, this is not observed in practice. We can see from Fig. 7 that the compared algorithms generate different sequences of iterates whose limit is the unique optimal solution of (6). We terminate each algorithm prematurely at a suboptimal solution, which varies based on our choice of algorithm.

**Table 3** Norm difference $\|\boldsymbol{\beta}^i_{APG} - \boldsymbol{\beta}^i_{ADMM}\|$, and cardinality of $\boldsymbol{\beta}^i_{APG}$ and $\boldsymbol{\beta}^i_{ADMM}$ for $i = 10, 50, 100, 500, 1000, 2500, 5000, 7500, 10000$. No iterates produced out-of-sample classification error for nearest centroid classification following projection onto $\boldsymbol{\beta}^i_{APG}$ and $\boldsymbol{\beta}^i_{ADMM}$

| Iteration | Norm Difference | Cardinality | |
|---|---|---|---|
| | | APG | ADMM |
| 10 | 0.1462 | 248 | 119 |
| 50 | 0.2468 | 240 | 78 |
| 100 | 0.2854 | 217 | 65 |
| 500 | 0.2414 | 103 | 51 |
| 1000 | 0.1464 | 70 | 49 |
| 2500 | 0.0187 | 52 | 49 |
| 5000 | 0.0103 | 49 | 47 |
| 7500 | 0.0062 | 49 | 47 |
| 10000 | 0.0027 | 48 | 47 |

To illustrate this phenomena, we recorded the iterates generated by Algorithm 1 using APG and ADMM with $\mu = 2$; we restricted our analyses to these methods to simplify our figures and similar behaviour would be observed using other algorithms for solving (6). We sampled balanced training and testing sets of $n = 200$ observations of dimension $p = 250$ from $N(\boldsymbol{\mu_1}, \boldsymbol{\Sigma})$ and $N(\boldsymbol{\mu_2}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu_i}$ is defined according to (27) and $\boldsymbol{\Sigma}$ is constructed as in the previous sections. We recorded the iterates $\boldsymbol{\beta}^i_{APG}, \boldsymbol{\beta}^i_{ADMM}$ generated by each of APG and ADMM (with $\mu = 2$) following $i = 10, 50, 100, 500, 1000, 2500, 5000, 7500, 10000$ iterations for solving (6) with regularization parameters $\gamma = 10^{-3}$, $\boldsymbol{\Omega} = \boldsymbol{I}$, and $\lambda = 0.05\bar{\lambda}$. We report the norm difference $\|\boldsymbol{\beta}^i_{APG} - \boldsymbol{\beta}^i_{ADMM}\|$, cardinality of $\boldsymbol{\beta}^i_{APG}$ and $\boldsymbol{\beta}^i_{ADMM}$, and out-of-sample classification error for nearest centroid classification following projection onto $\boldsymbol{\beta}^i_{APG}$ and $\boldsymbol{\beta}^i_{ADMM}$ for each value of $i$ in Table 3. We round any entry of a discriminant vector with magnitude less than $10^{-8}$ to 0 for the purposes of calculating cardinality. The calculated discriminant vectors are qualitatively similar but still differ significantly, particularly in cardinality. Specifically, the discriminant vectors generated by ADMM are much sparser than those calculated by APG, particularly early in the iterative process. This suggests that ADMM is converging to the unique optimal solution of (6) more quickly than APG for this particular data set and choice of augmented Lagrangian penalty parameter $\mu$; we investigate the sensitivity of ADMM to the choice of $\mu$ further in Sect. 4.7.1. However, we should also note that both methods converge to essentially identical solutions within 10000 iterations. We should also note that all of the discriminant vectors yield classifiers with zero out-of-sample classification errors.

### 4.4 Scaling experiments

We next performed a series of simulations to investigate the relationship between the performance of our algorithms and the number of features in the underlying data set. For each

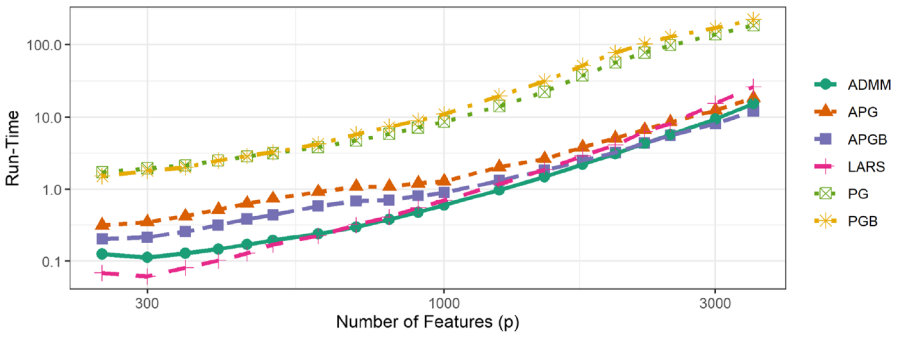
$$p \in \{250, 300, \ldots, 500, 600, \ldots, 1000, 1250, \ldots, 2500, 3000, 3500\},$$

we sample 50 data sets, containing two classes drawn from the Gaussian distributions as described in Sect. 4.1. For each value of $p$, we sample $\lceil p/10 \rceil$ training and testing observations from each class. Items in each class are sampled from a multivariate Gaussian distribution with means $\boldsymbol{\mu_1}, \boldsymbol{\mu_2} \in \mathbf{R}^p$ defined by (27). Both class distributions have covariance matrix $\boldsymbol{\Sigma}$ having diagonal entries equal to 1 and off-diagonal entries equal to 0.75.

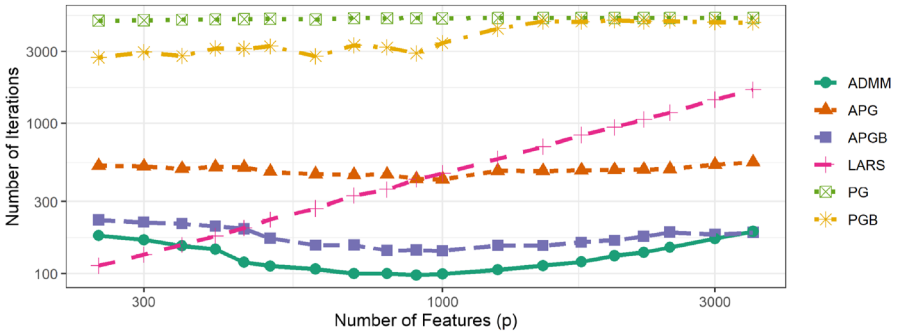
We apply nearest centroid classification following projection onto the approximate solution of (2) given by the proximal gradient method, accelerated proximal gradient method (with and without backtracking line search) (PG, PGB, APG, APGB), alternating direction method of multipliers (ADMM), and least angle regression method (LARS). We perform exactly one full iteration of the block coordinate descent method (Algorithm 1) for each $\boldsymbol{\beta}$-subproblem solver; as before, we should expect Algorithm 1 to terminate after one full iteration, since the optimal choice of $\boldsymbol{\theta}$ is obtained in the first iteration (in the absence of numerical error). We choose the regularization parameters in (2) to be $\gamma = 10^{-3}$, $\boldsymbol{\Omega} = \boldsymbol{I}$, and choose $\lambda$ from $\bar{\lambda}/2^c$, $c \in \{3, 2, 1, 0, -1\}$, where $\bar{\lambda}$ is defined as in (25) using 5 fold cross validation. We choose the value of $\lambda$ with minimum number of classification errors among those which generated discriminant vectors with at most $0.025p$ nonzero features. We used the augmented Lagrangian penalty parameter $\mu = 2$ in ADMM. We use the stopping tolerance $10^{-4}$ and a maximum of 1000 iterations during the cross validation phase. We used backtracking parameters $\bar{L} = 0.25$ and $\eta = 1.25$ in each run. After $\lambda$ is chosen via cross validation, we solved (2) using the full training set. We terminated the proximal methods when their stopping condition is met with tolerance $10^{-4}/\sqrt{p}$ or 5000 subproblem iterations have been performed. We use a less strict stopping tolerance than earlier analyses to minimize the number of iterations performed when $p$ is large, and, thus, decreasing overall run-time of the experiment.

The LARS heuristic was terminated after a solution was obtained containing $0.25p$ nonzero features or the convergence criterion is satisfied with tolerance $10^{-4}/\sqrt{p}$. The decision to choose $\lambda$ via cross validation rather than a specific value (as in the previous section) was made to account for the fact that each call of the LARS heuristic calculates a full regularization path for (2). We perform cross validation to choose $\lambda$ to more accurately compare the computational resources needed by PG, PGB, APG, APGB, ADMM, and LARS to calculate a full regularization path.

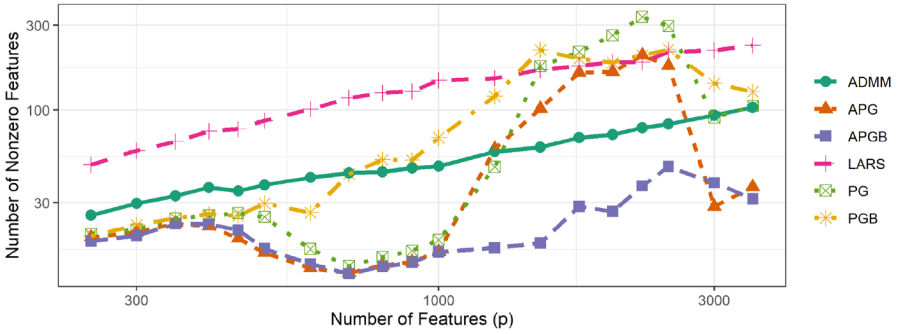
Figure 8 summarizes the results of these simulations. We note that the accelerated proximal gradient and alternating direction method of multipliers consistently outperform the traditional LARS method in terms of run-time and number of iterations performed with $p$ is large. In particular, these methods require significantly fewer iterations and terminate in less time than LARS for $p > 1000$. The approximate slopes of the plots of average run-times indicate that this phenomena will only be amplified as we increase $p$ further, since the slope of the curve for LARS exceeds that of APG, APGB, and ADMM. This largely agrees with the phenomena predicted by the operation counts discussed in Sect. 3.4. On the other hand, the proximal gradient methods (PG, PGB) typically do not converge within the maximum number of iterations, which undermines any improvements to computational complexity due to their relatively inexpensive iterations.




(a) Time




(b) Number of iterations




(c) Cardinality

**Fig. 8** Average run-time (in seconds), number of iterations performed, and cardinality of returned solution plotted as a function of number of features $p$; values of each statistic were averaged across 50 trials. All axes use logarithmic scale

We should note that we expect the cardinality of our obtained discriminant vectors to increase as a function of $p$, since the size of the blocks of entries with elevated values in the class-means $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$ grows linearly with $p$. This agrees with the plotted curves in Fig. 8c. This also explains the linear increase in number of iterations before termination of the LARS method, since the number of iterations depends on the desired number of nonzero entries; in turn, this, along with increase in per-iteration

cost as $p$ increases, explains the increase in total run-time of LARS as $p$ increases. Finally, the cardinality of returned discriminant vectors scales similarly for the four proximal gradient methods (PG, PGB, APG, APGB) and LARS. The discriminant vectors returned by ADMM consistently contain fewer nonzero entries than the four other methods, which agrees with the behaviour observed in the Sect. 4.2.

This comparison is somewhat unfair to the LARS heuristic since the number of nonzero discriminant vector features, which should correlate with the number of large magnitude entries of the difference of means $\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2$, is increasing linearly with $p$. Thus, we seek relatively dense discriminant vectors, scaling linearly with $p$. This is exactly the situation where we should expect ADMM and APG to be more efficient than LARS according to (23).

However, if the number of nonzero predictor variables is constant or grows relatively slowly as $p$ increases, we should expect LARS to be more efficient than APG and ADMM (again, according (23)). To confirm this empirically, we repeated this experiment but chose the mean vectors $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ to differ from zero in the first 100 and second 100 features, respectively, for all values of $p$. We stopped the LARS algorithm when an iterate with at least 200 nonzero features was found or stopping tolerance $10^{-4}/\sqrt{p}$ is met; we then chose the iterate with minimum out-of-sample classification rate for our discriminant vector. All other experimental settings were kept identical to that in the previous discussion. We omit the unaccelerated proximal gradient methods PG/PGB since the previous analysis established that they are significantly less than the other methods under comparison.

The results of this trial can be found in Fig. 9. As expected, LARS tends to more efficient than APG and ADMM in this setting. This suggests that one must be careful to consider problem dimension and the underlying properties of the desired classifiers (e.g., sparsity) when choosing a heuristic for solving (6). Specifically, if $p$ is small or we seek a sparse discriminant vector with relatively few nonzero entries, then we should use LARS to solve (6); otherwise, we should favor APG or ADMM.

### 4.5 Classification of real-world data

We performed similar analyses using data sets drawn from the UC Riverside Time-Series Clustering and Classification data repository Dau et al. (2018) to verify that the behaviour observed with synthetic data is also observed when classifying real-world data. We applied each of the methods APG, ADMM, SZVD, and LARS to learn classification rules for each of the data sets in the UCR repository with number of training samples $n$ less than the number of predictive features $p$; this yielded a collection of 63 data sets to analyze. We omit PG and the backtracking methods from this analysis because our analysis of synthetic data established that these methods are typically less effective than the remaining approaches. We use each remaining sparse discriminant analysis heuristic to obtain $q = K - 1$ sparse discriminant vectors and then perform nearest-centroid classification after projection onto the subspace spanned by these discriminant vectors.

In all experiments, we set $\gamma = 10^{-3}$ and $\boldsymbol{\Omega}$ to be the $p \times p$ identity matrix $\boldsymbol{\Omega} = \boldsymbol{I}$. We choose $\lambda$ using 5 fold cross validation from the set of potential $\lambda$ of the

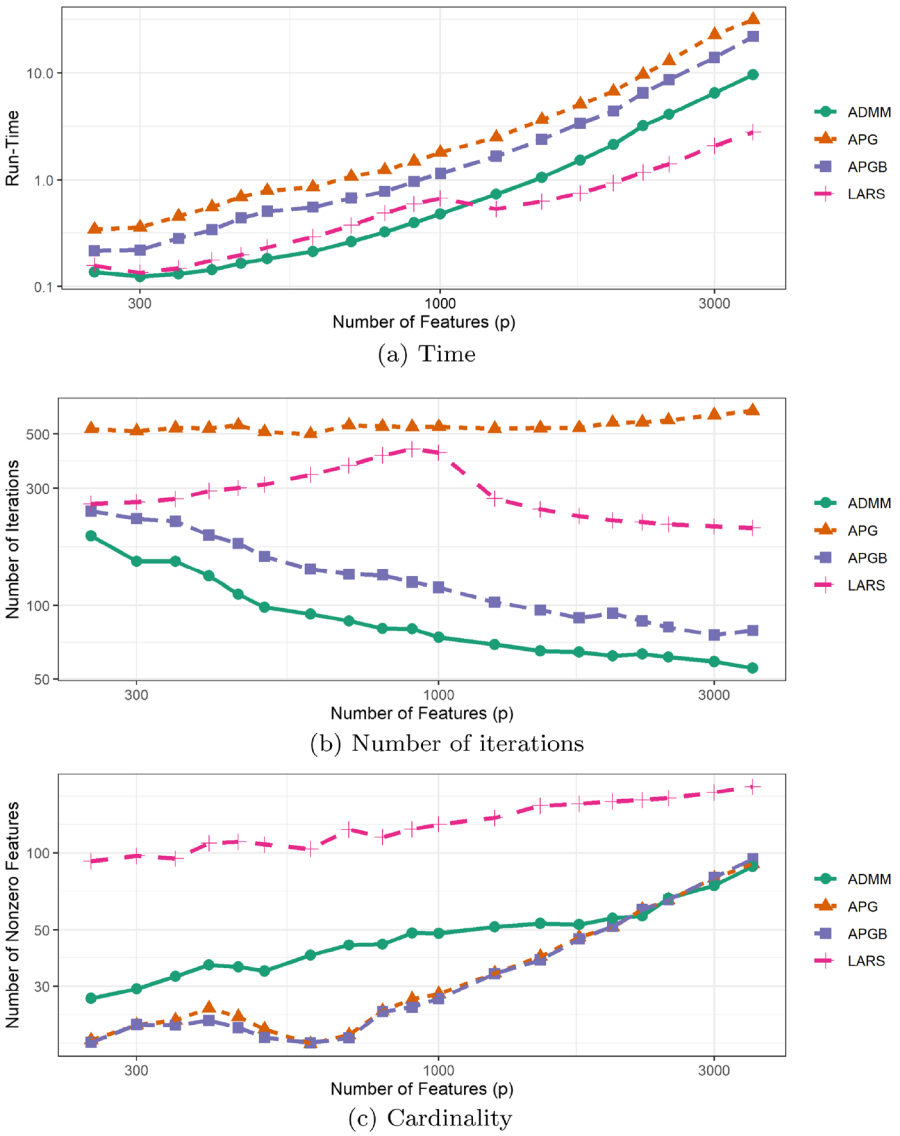



**Fig. 9** Average run-time (in seconds), number of iterations performed, and cardinality of returned solution for SOS problem with fixed number of nonzero entries of $\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2$ (200) plotted as a function of number of features $p$. Values of each statistic were averaged across 50 trials. All axes use logarithmic scale

form $\bar{\lambda}/2^c$ for $c = 3, 2, 1, 0, -1$ with $\bar{\lambda}$ defined by (25) to be the value of $\lambda$ with fewest average number of misclassification errors over training-validation splits amongst all $\lambda$ which yield discriminant vectors containing at most 25% nonzero entries. During the cross validation stage, we terminate each proximal algorithm in the inner loop after 1500 iterations or a $10^{-4}$ suboptimal solution is obtained. After $\lambda$ is chosen via cross validation we solve (2) using the full training data set. Here we terminate each proximal

algorithm in the inner loop after 2000 iterations or a $10^{-6}$ suboptimal solution is found and the outer loop is stopped after one iteration if $K = 2$ or a maximum number of 250 iterations or a $10^{-3}$ suboptimal solution has been found otherwise. The augmented Lagrangian parameter $\mu = 2$ was used in ADMM.

We calculate the full regularization path for $\lambda$ using the LARS algorithm, choosing the set of discriminant vectors from the full path with maximum in-sample classification accuracy.[4] We terminate each call to LARS in the inner loop after a solution with $0.25p$ nonzero entries is found, or stopping tolerance $10^{-6}$ is met, or 3000 iterations are performed.

We use the augmented Lagrangian parameter $\beta = 5$ and choose the regularization parameter $\gamma$ in SZVD from the exponentially spaced grid $\bar{\gamma}/2^c$ for $c = 3, 2, 1, 0, -1$, with $\bar{\gamma}$ defined by (26) using 5 fold cross-validation; $\gamma$ is chosen to minimize average validation set misclassification error amongst all sets of discriminant vectors with at most 35% nonzero entries. We stop SZVD after a maximum of 250 iterations or a solution satisfying the stopping tolerance of $10^{-5}$ is obtained. These parameters were chosen experimentally to ensure that all methods converge for each data set in the benchmarking set. It is likely that some variation in performance of the heuristics across data sets could be eliminated by more carefully tuning parameters for each individual data set. We assigned the same choice of parameters for each experiment to avoid having to tune parameters separately for all 63 data sets.

#### 4.5.1 Comparison of classification accuracy

To empirically test accuracy of each proposed classification heuristic, we calculated the out-of-sample misclassification rate for each data set in the UCR repository. We include baseline accuracies based on the classification results of 1-Nearest Neighbor classifiers: each test observation was assigned the class label of its nearest training observation. As a measure of similarity, we use both *Euclidean distance* (ED) and *dynamic time warping distance* with fixed warping window width $w = 100$ (DTW) and learned window width (DTWL). The out-of-sample misclassification rate for each of these classifiers is provided by the UCR Time Series Archive; we direct the reader to refer to (Dau et al. 2019, Section II) for further details.

For each ordered pair of classification heuristics, we perform a one-sided Wilcoxon signed-rank test. Specifically, for each pair of classification heuristics $(i, j)$ we perform a one-sided Wilcoxon test to test the null hypothesis $H_0 : \text{err}(i) = \text{err}(j)$ against the alternative hypothesis $H_a : \text{err}(i) < \text{err}(j)$, where $\text{err}(x)$ denotes the population average misclassification error rate for classifier $x$. Figure 12 provides a box plot visualizing average misclassification rate for each method, as well as a table of $p$-values for the one-sided Wilcoxon significance tests. We observe a significant difference between our APG and ADMM classifiers and nearest neighbors classifiers using dynamic time warping distance with learned window width (DTWL) and fixed

[4] This choice of method of tuning $\lambda$ differs from that used in earlier versions of this manuscript, where we chose $\lambda$ via cross-validation or based on out-of-sample classification rate. The results of this set of analyses largely agree with those of the earlier manuscripts, except with significantly decreased run-times for LARS when compared to the approach applying cross-validation, and modestly increased misclassification error when compared to those trained using out-of-sample accuracy.

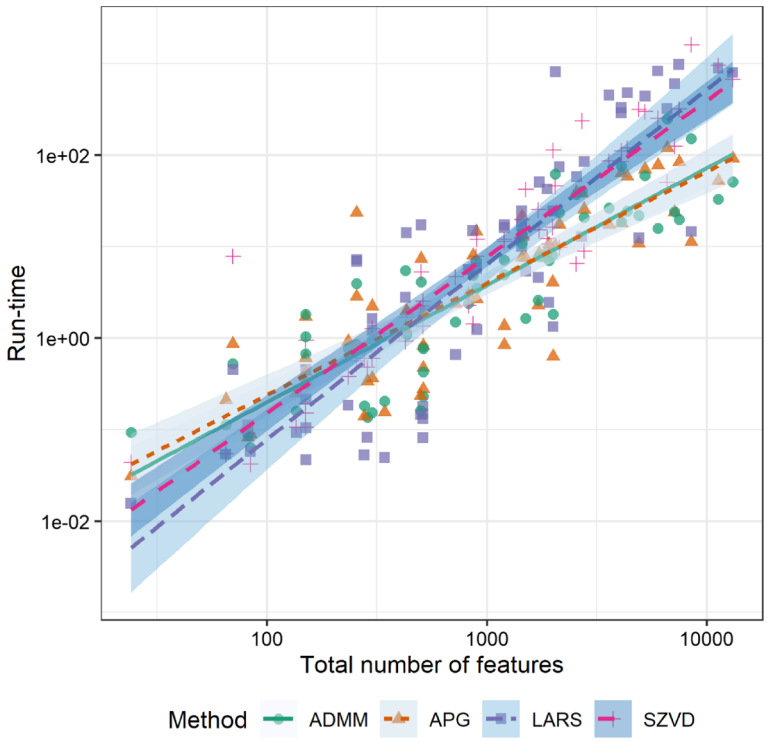
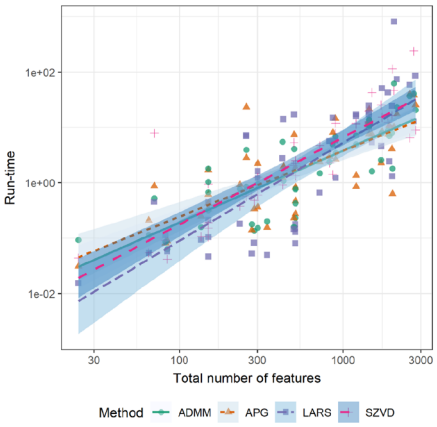


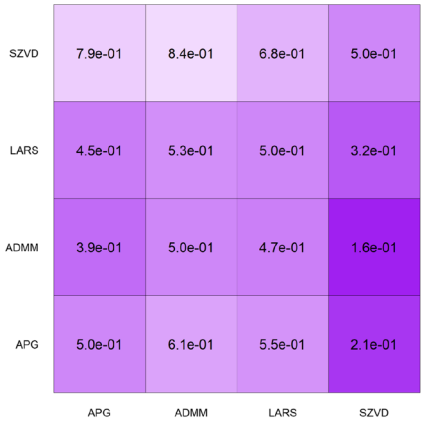

**Fig. 10** Plot of run-time in seconds of each sparse discriminant heuristic as a function of the total number of predictive features ($p$ times number of discriminant vectors). We also fit a line to the set of run-times for each method and 95%-confidence intervals for these linear models. Both axes use logarithmic scale

length. We also observe evidence that APG and ADMM are, on average, less accurate than the LARS classifier ($p$-values 0.048 and 0.026, respectively). On the other hand, the results of these hypothesis tests suggests a significant improvement in accuracy when using the DTWL classifier over all classifiers except the DTW classifier, and all methods provide improved accuracy over SZVD.

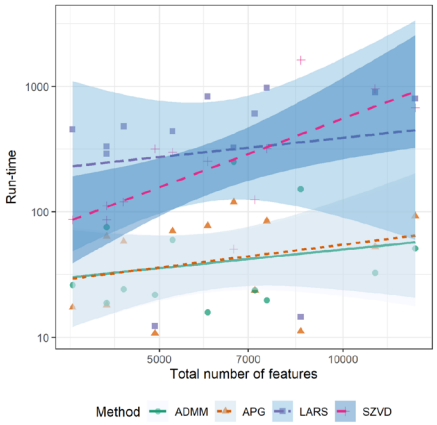
The observed differences between the accuracy of nearest neighbors classifiers, particularly those using dynamic time warping distances, and SOS classifiers can be partially explained by the extremely poor accuracy of SOS classifiers for a limited number of data sets. Linear discriminant analysis-based classifiers are only applicable under the assumption that data is linearly separable following projection onto a lower dimensional subspace and that data from all classes are sampled from distributions with shared covariance matrix. The poor accuracy of the SOS classifiers suggest that these assumptions are not satisfied by this subset of the benchmarking repository. If we omit the data sets in the UCR repository for which the APG classifier yields a misclassification rate of at least 60%, we obtain the average misclassification rates and attained significance visualized in Fig. 13. After restricting the benchmarking data set in this way, we observe no significant difference between the APG-trained



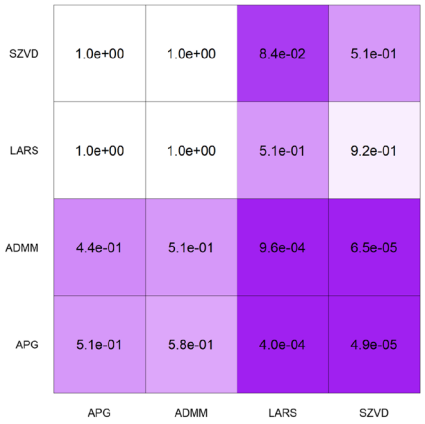

(a) Run-times for Small-scale Data




(b) Hypothesis Tests for Small-scale Data




(c) Run-times for Large-scale Data

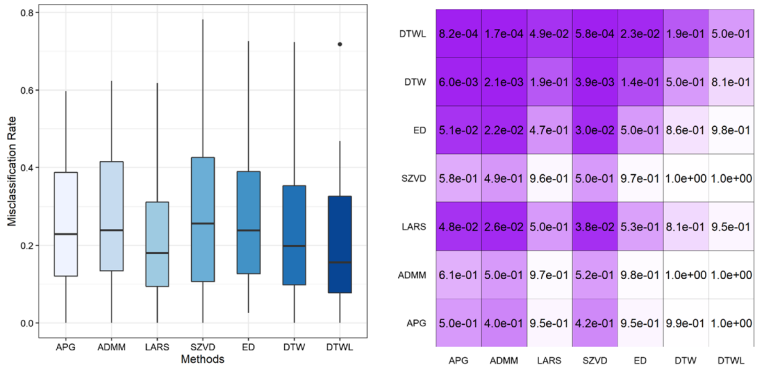



(d) Hypothesis Tests for Large-scale Data

**Fig. 11** Plots of run-time for small-scale (total number of features less than 3000) and larger-scale data (total number of features exceeding 3000) included in the UCR benchmarking repository. We also include the results of from one-sided Wilcoxon signed-rank tests comparing computational efficiency of each pair of sparse discriminant analysis heuristics for each subset of benchmarking data. The $(i, j)$ box represents the observed $p$-value for the test with null hypothesis $H_0 : \text{time}(i) \geq \text{time}(j)$ and alternative hypothesis $H_a : \text{time}(i) < \text{time}(j)$ where $\text{time}(x)$ denotes the expected total run-time of heuristic $x$ on a given data set; darker colors correspond to smaller $p$-values or higher significance

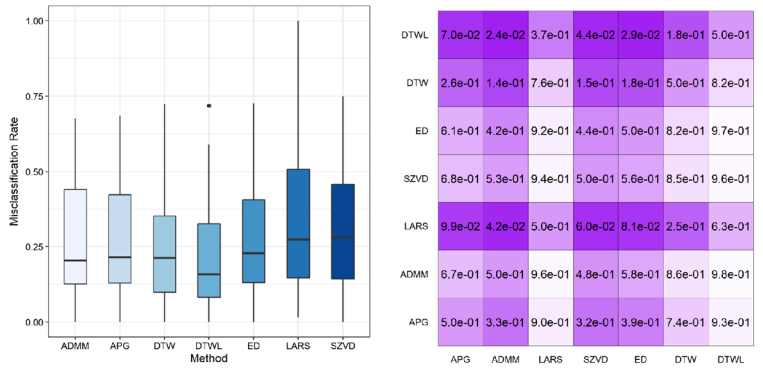
classifiers and the nearest neighbor classifier DTWL or the LARS-trained classifier at a significance level of $p < 0.05$.

We note that we do not consider this reduced benchmarking set in an attempt to overstate the classification accuracy of the proposed methods APG and ADMM relative to DTW-based nearest neighbors methods. Instead, we want to emphasize that the average difference in accuracy is due to the relatively poor performance of SOS models for a modest number of problems, for which linear classifiers are possibly ill-suited, rather than the SOS models performing poorly on average over all benchmarking




(a) Misclassification Rate (b) Results of Hypothesis Tests for Accuracy

**Fig. 12** Box plots of out-of-sample misclassification rates. We also include box plots for misclassification rate for nearest neighbor classification using Euclidean distance (ED) and Dynamic Time Warping distance with fixed warping constraint parameter $w = 100$ (DTW), and learned $w$ (DTWL). We also plot results of one-sided Wilcoxon signed-rank tests for misclassification rate. The $(i, j)$ box represents the observed $p$-value for the test with null hypothesis $H_0 : \text{err}(i) \geq \text{err}(j)$ and alternative hypothesis $H_a : \text{err}(i) < \text{err}(j)$ where $\text{err}(x)$ denotes the expected fraction of misclassified test observations by classification heuristic $x$




(a) Misclassification Rate with Data Sets Omitted (b) Results of Hypothesis Tests for Accuracy with Data Sets Omitted

**Fig. 13** Box plots and attained significance/$p$-values of out-of-sample misclassification rates with data sets with error at least 70% omitted. Here, we observe no significance difference in classification error ($p$-value less than 0.05) between APG/ADMM classifiers and DTWL

data sets. This suggests that SOS classifiers, including those trained using the LARS algorithm, exhibit comparable classification performance to the baseline provided by nearest neighbors classifiers when restricted to instances where their use is appropriate, i.e., their underlying statistical assumptions are met.

### 4.5.2 Comparison of run-times

In terms of computational complexity, we can observe two general trends: LARS is consistently more efficient than APG and ADMM when the total number of predictor variables, $qp$, is small (say, $qp < 1000$), and APG/ADMM are significantly more

efficient than LARS when the number of features is moderate to large ($qp \geq$ '000). Figure 10 plots the total run-time of each sparse discriminant heuristic (including training of parameters by cross-validation), along with linear models fit to observed run-times. There are clear bifurcation points between $500 < qp < 1000$, where the linear models for run-time of APG and ADMM cross that of LARS. To investigate this phenomena further, we isolated run-times for data sets with $qp < 3000$ and $qp \geq 3000$ and performed one-sided Wilcoxon tests for the null hypothesis $H_0 : \text{time}(i) = \text{time}(j)$ and alternative hypothesis $H_a : \text{time}(i) < \text{time}(j)$ under both settings. The results of these significance tests can be found in Fig. 11, along with plots of run-times. These tests strongly suggest that both APG and ADMM require significantly less computation than LARS when the total number of predictor variables is greater than 3000: we observe $p$ values on the order of $10^{-4}$ when testing $H_0 : \text{time(APG)} = \text{time(LARS)}$ and $H_0 : \text{time(ADMM)} = \text{time(LARS)}$. On the other hand, we see modest evidence that LARS is more efficient than both ADMM and APG when the total number of predictor variables is less than 3000, but not at a statistically significant level. We should also note that the computational cost of the LARS algorithm is at least partially inflated by the fact that we seek somewhat dense discriminant vectors (with terminating cardinality $0.25p$), although LARS regularly terminated with sparse optimal solutions, obtained in fewer than $0.25p$ iterations. Finally, all methods were more efficient than the SZVD method. This scaling of computational cost largely agrees with that observed for synthetic data in Sect. 4.4 and predicted by (23).

We should note that we do not include nearest neighbor classifiers in this discussion of computational efficiency, although we used these methods to obtain a baseline accuracy to benchmark our proposed methods against. The accuracies of these classifiers were provided with the UCR repository and, thus, did not require retraining of the classifiers. Naive implementation of nearest neighbors methods requires at least $\mathcal{O}(n^2 p)$ operations to calculate pairwise Euclidean distances and $\mathcal{O}(n^2 p^2)$ flops to calculate DTW distances (plus additional operations for training the window width $w$); optimized methods for DTW reduce this computational cost to $\mathcal{O}(n^2 p)$ flops. Therefore, we expect our methods to scale at least as well as these nearest neighbors methods.

### 4.6 Multispectral X-ray images and Ω of varying rank

To demonstrate the improvement in run-time obtained by using a low-rank $\boldsymbol{\Omega}$ in the elastic-net penalty, we perform pixelwise classification on multispectral X-ray images, as presented in Einarsson et al. (2017). The multispectral X-ray images are scans of food items, where each pixel contains 128 measurements (channels) corresponding to attenuation of X-rays emitted at different wavelengths (see Fig. 14). The measurements in each pixel thus give us a profile for the material positioned at that pixel's location (see Fig. 15).

We start by preprocessing the scans as in Einarsson et al. (2017) in order to remove scanning artifacts and normalize the intensities between scans. We scale the measurements in each pixel by the 95% quantile of the corresponding 128 measurements instead of the maximum. This scaling approach is more robust in the sense that it is less

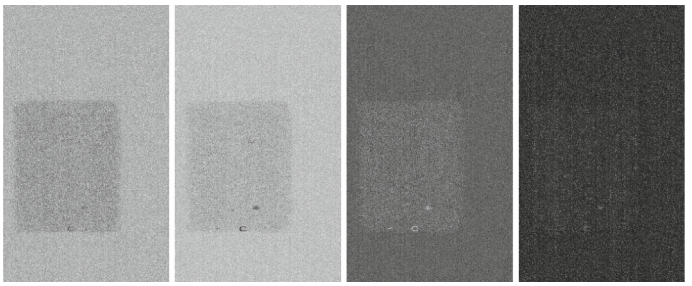


**Fig. 14** Grayscale images of different channels from a minced meat sample generated with a multispectral X-ray scanner after all preprocessing. From left to right are channels 2, 20, 50 and 100. The contrast decreases the higher we go in the channels and the variation in the measurements increases. Some foreign objects can be seen as small black dots

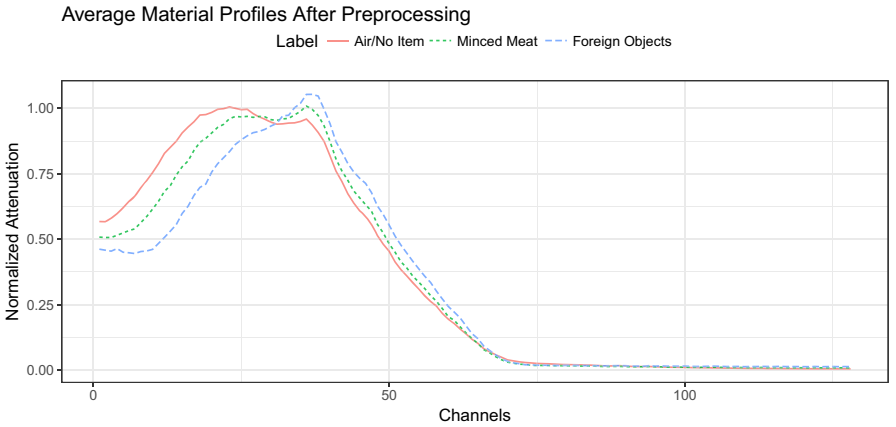



**Fig. 15** Profiles of materials seen in Fig. 14 over the 128 channels. The profile for each type of material, displayed here, is averaged over 500 pixels

sensitive to outliers compared to using the maximum. We create our training data by manually selecting rectangular patches from six scans. We have three classes, namely *background*, *minced meat* and *foreign objects*. We further subsample the observations to have balanced number of observations, where the class *foreign objects* was under represented. In the end we have 521 observations per class, where each observation corresponds to a single pixel. This data was used to generate Fig. 15. For training we use 100 samples per class, and the rest is allocated to a final test set. This process yields 128 variables per observation, but in order to get more spatially consistent classification, we also include data from the pixels located above, to the right, below and to the left of the observed pixel. Thus we have $p = 5 \cdot 128 = 640$ variables per observation. The measurements corresponding to our observation are thus indexed according to spatial and spectral position, i.e., observation $\mathbf{x}_i$ has measurements $x_{ijk}$, where $j \in \{0, 1, 2, 3, 4\}$ indicates which pixel the measurement belongs to (*center, above, right, bottom, left*), and $k \in \{1, 2, ..., 128\}$ indicates which channel.

We can impose priors according to these relationships of the measurements in the $\boldsymbol{\Omega}$ regularization matrix. We assume that the errors should vary smoothly in space and thus impose a Matérn covariance structure on $\boldsymbol{\Omega}^{-1}$ Matérn (2013):

$$C_\nu(d) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left( \sqrt{2\nu} \frac{d}{\rho} \right)^\nu K_\nu \left( \sqrt{2\nu} \frac{d}{\rho} \right). \tag{28}$$

The Matérn covariance structure (28) is governed by the distance $d$ between measurements. In (28), $\Gamma$ refers to the gamma function and $K_\nu$ is the modified Bessel function of the second kind. For this example we assume that all parameters are 1, except that $\nu$ is 0.5. We further assume that the distance between measurements $x_{ijk}$ and $x_{ij'k'}$ from observation $i$ is the Euclidean distance between the points $(x_j, y_j, z_k)$ and $(x_{j'}, y_{j'}, z_{k'})$, where $x_j, y_j, x_{j'}, y_{j'} \in \{-1, 0, 1\}$ and $z_k, z_{k'} \in \{1, 2, ..., 128\}$. The distance is thus the same as in the image grid (center, top, bottom, left, right pixel location), and $z$-dimension corresponds to the channel.

We use a stopping tolerance of $10^{-5}$ and a maximum of 1000 iterations for the inner loop using the accelerated proximal algorithm, and a stopping tolerance of $10^{-4}$ and maximum 1000 iterations for the outer block-coordinate loop. The regularization parameter for the $l_1$-norm is selected as $\lambda = 10^{-3}$ and $\gamma = 10^{-1}$ for the Tikhonov regularizer. We present the run-time for varying $r$ in Fig. 16 and the accuracy with respect to varying $r$ in Fig. 17. There is a clear linear trend in rank $r$ for the increase in run-time; this agrees with the analysis of Sect. 3.4. We also estimate the accuracy for a identity regularization matrix, i.e., $\boldsymbol{\Omega} = \boldsymbol{I}$, with the same regularization parameters $\gamma$ and $\lambda$ and achieve accuracy of 0.948, which is approximately the same accuracy as when using $\boldsymbol{\Omega}^{400}$. To demonstrate the effect that the rank of $\boldsymbol{\Omega}$ has on computational complexity, we obtain the singular value decomposition of $\boldsymbol{\Omega} = \sum_{i=1}^{p} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$, and construct a low-rank approximation to $\boldsymbol{\Omega}$ using the first $r$ singular vectors and singular values: $\boldsymbol{\Omega}^{\boldsymbol{r}} = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$. We supplied the same parameters to the function `sda` from the library `sparseLDA`; `sda` required 267 s to run and achieved an accuracy of 0.949. The maximum accuracy is achieved with the full regularization matrix, which is 0.957.

### 4.7 Summary

Our proposed proximal methods for sparse discriminant analysis provide a decrease in classification error over the existing LARS approach in almost all experiments. Moreover, we see a significant improvement in terms of computational resources used by the accelerated proximal gradient method (APG and APGB) and alternating direction method of multipliers (ADMM) over LARS in medium to large-scale problem instances, i.e., when the number of predictor variables $p$ exceeds 1000, without significant loss of classification accuracy when compared to standard nearest neighbors classifiers (ED, DTW); we remind the reader that SOS with APG and ADMM does not match the classification accuracy of the nearest neighbor classifier using dynamic time wapring distance with learned window width (DTWL) due to poor classification performance of SDA for a small number of data sets for which SDA does not seem

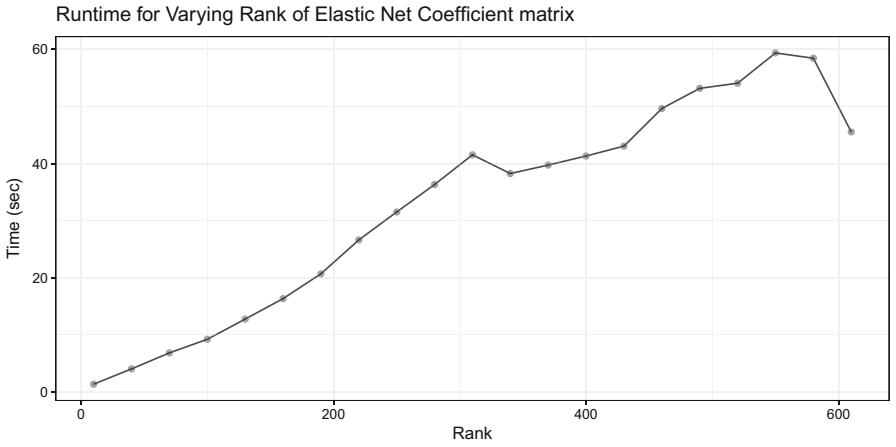



**Fig. 16** Run-time as function of rank($\boldsymbol{\Omega}$). The run-time also includes the creation of the low-rank approximated $\boldsymbol{\Omega}$ matrix

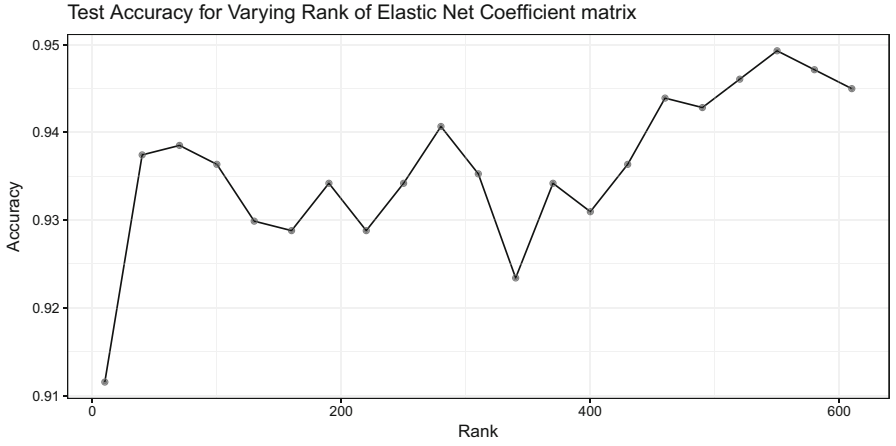



**Fig. 17** Test accuracy as function of rank($\boldsymbol{\Omega}$)

applicable, and that our methods are more efficient than DTWL, offer an element of feature selection via sparsity, and are amenable to learning tasks other than classification as a general dimension reduction tool. We should note that this agrees with the theoretical estimates of computational cost of these methods given in Sect. 3.4 and "Appendix A". Specifically, both APG and ADMM converge linearly with per-iteration complexity on the order of $\mathcal{O}(p)$ floating point operations per-iteration, which leads to overall computation time, as measured in floating point operations, to be far less than the $\mathcal{O}(p^3)$ flops of the classical LARS method. In our experiments, the decrease in run-time is most significant when $p$ is large, where the cost of $\mathcal{O}(p^3)$ flops for LARS becomes prohibitive.

It is important to note that the slow convergence of the proximal gradient method (PG/PGB) without acceleration yields significantly longer run-times despite the

decreased per-iteration cost. Finally, we note that there appears to be limited benefit from the use of backtracking line search, when compared to a constant step size given by the Frobenius norm estimate $\|\boldsymbol{A}\|_F$ of the Lipschitz constant. Specifically, the results of these experiments indicate that using a constant step length yields similar classification performance to the backtracking approach, but without a significant increase in run-time due to repeated calculation of $\nabla f$.

#### 4.7.1 Comparison of ADMM and APG

The results of our empirical analysis suggest that the use of either APG and ADMM to solve (6) may yield a significant improvement over the classical LARS-EN heuristic, particularly when $p$ is large and we seek dense discriminant vectors. However, which of these two methods is most efficient seems to vary under different experimental conditions. Specifically, APG generally requires less overall run-time than ADMM when analyzing data sets from the UCR benchmarking repository considered in Sect. 4.5. On the other hand, ADMM tends to converge more quickly and require less computation than APG when analyzing synthetic data, particularly in our convergence tests (Sect. 4.2) and scaling tests (Sect. 4.4). This suggests that performance of the ADMM heuristic is sensitive to the choice of the augmented Lagrangian parameter $\mu$, as this is the only parameter that varies between these different analyses.

We performed the following analysis to further illustrate this sensitivity to the choice of $\mu$. We generated 100 different data sets containing $n = 200$ training observations sampled from each of the $p = 2000$ dimensional Gaussian distributions $N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$ and $N(\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$ as in Sect. 4.2. For each problem instance, we (approximately) solved (2) using APG and ADMM with $\mu \in \{1/25, 1/5, 1, 5, 25, 125\}$ to solve (6). We set $\gamma = 10^{-3}$, $\boldsymbol{\Omega} = \boldsymbol{I}$, $\lambda = 0.05\bar{\lambda}$ and terminate APG or ADMM if their respective stopping criteria are met with tolerance $10^{-4}/\sqrt{p}$. For each data set, we record the objective function value of (6) at each iteration, as well as cardinality of the obtained discriminant vector, number of iterations performed before termination, total run-time, and out-of-sample classification accuracy for a testing set of 200 observations drawn from each of $N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$ and $N(\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$. Note that we are essentially repeating our analysis from Sect. 4.2, except this time we focus only APG and ADMM under varying choices of $\mu$.

Figure 18 and Table 4 summarize the results of this analysis. Recall that ADMM follows an (approximate) dual ascent applied to the dual functional of (16). The augmented Lagrangian parameter $\mu$ controls emphasis between the objective function of (16) and the quadratic penalty function $\|\boldsymbol{x} - \boldsymbol{y}\|_2^2$. When $\mu$ is large, iterations of ADMM generally decrease disagreement between $\boldsymbol{x}$ and $\boldsymbol{y}$ while making only modest decreases, or even increases, in the objective $\frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} + \boldsymbol{d}^T\boldsymbol{x} + \lambda\|\boldsymbol{y}\|_1$; this corresponds to the slow convergence observed when $\mu = 125$. On the other hand, when $\mu$ is very small, we have significant decrease in the objective function each iteration, but many iterations are required before disagreement between $\boldsymbol{x}$ and $\boldsymbol{y}$ is small. We observe a two-stage phenomena in our experiments, where early iterations of ADMM feature slow decrease or increase in objective value while the gap between $\boldsymbol{x}$ and $\boldsymbol{y}$ decreases, followed by sharp descent in objective value; this initial period is longest when $\mu$ is small. This provides empirical evidence that we need to carefully choose the penalty

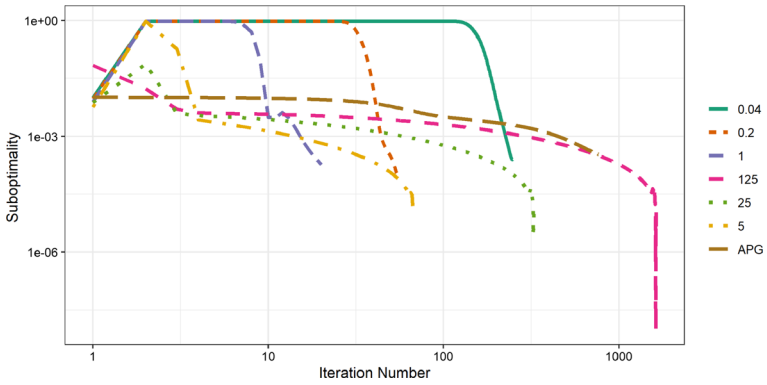



**Fig. 18** Difference of objective value and optimal value of each iterate averaged across 100 Gaussian data sets. We note that ADMM converges in fewer iterations than APG for all choices of $\mu$ except $\mu = 125$

**Table 4** Average number of nonzero entries (of $p = 2000$), average run-time in seconds, and average number of iterations performed before termination, with standard deviation in parentheses

| Method | Cardinality | Run-Time | Number of Iterations |
|---|---|---|---|
| APG | 536.1 (92) | 2.902 (0.531) | 766 (136.9) |
| $\mu = 1/25$ | 280.4 (7.2) | 0.941 (0.066) | 248.7 (3.4) |
| $\mu = 1/5$ | 291.3 (8.7) | 0.29 (0.022) | 55.9 (0.7) |
| $\mu = 1$ | 410.1 (12.7) | 0.171 (0.013) | 20.7 (0.6) |
| $\mu = 5$ | 426.6 (12.8) | 0.33 (0.026) | 67.5 (2.9) |
| $\mu = 25$ | 428.4 (12.3) | 1.212 (0.1) | 328.6 (14.6) |
| $\mu = 125$ | 428.7 (12.3) | 5.63 (0.475) | 1639.7 (73.4) |

We see that ADMM yields sparser solutions than APG for all choices of $\mu$ and is more efficient than APG for all $\mu$ except $\mu = 125$, in terms of both total run time and number of iterations performed. All methods achieved 100% out-of-sample classification rate for all 100 training/testing data sets

parameter $\mu$ in order to optimize convergence of our ADMM heuristic, and partially explains the gap in efficiency between APG and ADMM observed in our experiments. Specifically, ADMM is more efficient than APG only if a suitable choice of $\mu$ is used; if we do not carefully tune this penalty parameter, APG can be significantly more efficient than ADMM.

## 5 Conclusion

We have proposed new algorithms for solving the sparse optimal scoring problem for high-dimensional linear discriminant analysis based on block coordinate descent and proximal operator evaluations. We observe that these algorithms provide significant improvement over existing approaches for solving the SOS problem in terms of efficiency and scalability. These improvements are most acute in the case that specially structured Tikhonov regularization is employed in the SOS formulation; for exam-

ple, the computational resources required for each iteration scale linearly with the dimension of the data if either a diagonal or low-rank matrix is used.

Moreover, we establish that any convergent subsequence of iterates generated by one of our algorithms converges to a stationary point. Finally, numerical simulation establishes that our approach provides an improvement over existing methods for sparse discriminant analysis, particularly when the number of nonzero predictor variables in the discriminant vectors is relatively large.

These results present several exciting avenues for future research. Although we focus primarily on the solution of the optimal scoring problem under regularization in the form of a generalized elastic net penalty, our approach should translate immediately to formulations with any nonsmooth convex penalty function. That is, the framework provided by Algorithm 1 can be applied to solve the SOS problem (2) obtained by applying an arbitrary convex penalty to the objective of the optimal scoring problem (1). The resulting optimization problem can be approximately solved by alternately minimizing with respect to the score vector $\boldsymbol{\theta}$ using the formula (4) and with respect to the discriminant vector $\boldsymbol{\beta}$ by solving a modified version of (6). The proximal methods outlined in this paper can be applied to minimize with respect to $\boldsymbol{\beta}$ if the regularization function is convex; however it is unlikely that the computational resources necessary for this minimization will scale as favorably as with the generalized elastic net penalty. On the other hand, the convergence analysis presented in Sect. 2.2 extends immediately to this more general framework. Of particular interest is the modification of this approach to provide means of learning discriminant vectors for data containing ordinal labels, data containing corrupted or missing observations, and semi-supervised settings.

Finally, the results found in Sect. 2.2, as well as Appendices C and D establish that any convergent subsequence of iterates generated by our block coordinate descent approach must converge to a stationary point. However, it is still unclear when this sequence of iterates is convergent, or at what rate these subsequences converge; further study is required to better understand the convergence properties of these algorithms.

Similarly, despite the empirical evidence provided in Sect. 4, it is unknown what conditions ensure that data is classifiable using sparse optimal scoring and, more generally, linear discriminant analysis. Extensive consistency analysis is needed to determine theoretical error rates for distinguishing random variables drawn from distinct distributions.

**Supplementary Information** The online version contains supplementary material available at https://doi.org/10.1007/s11634-022-00530-6.

**Acknowledgements** We are grateful to Mingyi Hong for his helpful comments and suggestions, and to the anonymous reviewers, whose suggestions significantly improved this manuscript. This research was supported in part by University of Alabama Research Grant RG14678.

**Funding** B. Ames was supported in part by National Science Foundation Grants #20212554 and #2108645, University of Alabama Research Grants RG14678 and RG14838, and a UA Cyberseed Grant. G. Einarsson's PhD scholarship was funded by the Lundbeck foundation and the Technical University of Denmark. S. Atkins was part of the University Scholars Program at the University of Alabama and received a graduate student fellowship funded by the University of Florida while this research was conducted. This work was

made possible in part by a grant of high performance computing resources and technical support from the Alabama Supercomputer Authority.

## Appendix A: Detailed calculation of per-iteration complexity

The most expensive step of both the proximal gradient method (Algorithm 2) and the accelerated proximal gradient method (Algorithm 4) is the evaluation of the gradient $\nabla f$. Given a vector $\boldsymbol{\beta} \in \mathbf{R}^p$, the gradient at $\boldsymbol{\beta}$ is given by

$$\nabla f(\boldsymbol{\beta}) = \boldsymbol{A}\boldsymbol{\beta} = 2\left(\boldsymbol{X}^T\boldsymbol{X} + \gamma\boldsymbol{\Omega}\right)\boldsymbol{\beta} = 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} + 2\gamma\boldsymbol{\Omega}\boldsymbol{\beta}.$$

The product $\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta}$ can be computed using $\mathcal{O}(np)$ floating point operations (flops) by computing $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta}$ and then $\boldsymbol{X}^T\boldsymbol{y}$. On the other hand, the product $\boldsymbol{\Omega}\boldsymbol{\beta}$ requires $\mathcal{O}(p^2)$ flops for unstructured $\boldsymbol{\Omega}$. However, if we use a *structured* regularization parameter $\boldsymbol{\Omega}$ we can significantly decrease this computational cost. Consider the following examples:

- Suppose that $\boldsymbol{\Omega}$ is a diagonal matrix: $\boldsymbol{\Omega} = \text{Diag}(\boldsymbol{u})$ for some vector $\boldsymbol{u} \in \mathbf{R}^p_+$. Then the product $\boldsymbol{\Omega}\boldsymbol{\beta}$ can be computed using $\mathcal{O}(p)$ flops:
  $(\boldsymbol{\Omega}\boldsymbol{\beta})_i = u_i\beta_i$.
  Moreover, we can estimate the Lipschitz constant $\|\boldsymbol{A}\|$ for use in choosing the step size $\alpha$ by $\|\boldsymbol{A}\| \leq 2\gamma\|\boldsymbol{\Omega}\| + 2\|\boldsymbol{X}\|_F^2 = 2\gamma\|\boldsymbol{u}\|_\infty + 2\|\boldsymbol{X}\|_F^2$, which requires $\mathcal{O}(np)$ flops, primarily to compute the norm $\|\boldsymbol{X}\|_F^2$.
- If the use of diagonal $\boldsymbol{\Omega}$ is inappropriate, we could store $\boldsymbol{\Omega}$ in factored form $\boldsymbol{\Omega} = \boldsymbol{R}\boldsymbol{R}^T$ where $\boldsymbol{R} \in \mathbf{R}^{p\times r}$, and $r$ is the rank of $\boldsymbol{\Omega}$. In this case, we have $\boldsymbol{\Omega}\boldsymbol{\beta} = \boldsymbol{R}(\boldsymbol{R}^T\boldsymbol{\beta})$, which can be computed at a cost of $\mathcal{O}(rp)$ flops. Thus, if we use a low-rank parameter $\boldsymbol{\Omega}$, say $r \leq \mathcal{O}(n)$, we can compute the gradient using $\mathcal{O}(np)$ flops. Similarly, we can estimate the step size $\alpha$ using $\|\boldsymbol{A}\| \leq 2\|\boldsymbol{R}\|_F^2 + 2\|\boldsymbol{X}\|_F^2$ (computed at a cost of $\mathcal{O}(rp + np)$ flops).

In either case, using a diagonal $\boldsymbol{\Omega}$ or low-rank factored $\boldsymbol{\Omega}$, each iteration of the proximal gradient method or the accelerated proximal gradient method requires $\mathcal{O}(np)$ flops. Similar improvements can be made if $\boldsymbol{\Omega}$ is tridiagonal, banded, sparse, or otherwise nicely structured.

Similarly, the use of structured $\boldsymbol{\Omega}$ can lead to significant improvements in computational efficiency in our ADMM algorithm. The main computational bottleneck of this method is the solution of the linear system in the update of $\boldsymbol{x}$:

$$(\mu\boldsymbol{I} + \boldsymbol{A})\boldsymbol{x}^{i+1} = \boldsymbol{d} + \mu\boldsymbol{y}^i - \boldsymbol{z}^i.$$

Without taking advantage of the structure of $\boldsymbol{A}$, we can solve this system using a Cholesky factorization preprocessing step (at a cost of $\mathcal{O}(p^3)$ flops) and substitution to solve the resulting triangular systems (at a cost of $\mathcal{O}(p^2)$ flops per-iteration). However, we can often use the Sherman-Morrison-Woodbury Lemma to solve this system more efficiently using the structure of $\boldsymbol{A}$. Indeed, fix $t$ and let $\boldsymbol{b} = \boldsymbol{d} + \mu\boldsymbol{y}^i - \boldsymbol{z}^i$. Then we

**Table 5** Upper bounds on floating point operation counts for most time consuming steps of each algorithm

| | | Diagonal $\boldsymbol{\Omega}$ | Rank $r$ $\boldsymbol{\Omega}$ | Full rank $\boldsymbol{\Omega}$ |
|---|---|---|---|---|
| Proximal Gradient | $\nabla f$ | $\mathcal{O}(np)$ | $\mathcal{O}(rp + np)$ | $\mathcal{O}(p^2)$ |
| | Bound on $\|\boldsymbol{A}\|$ | $\mathcal{O}(np)$ | $\mathcal{O}(rp + np)$ | $\mathcal{O}(p^2 \log p)$ |
| ADMM | $(\mu \boldsymbol{I} + \boldsymbol{A})\boldsymbol{x} = \boldsymbol{b}$ | $\mathcal{O}(n^3 + n^2 p)$ | $\mathcal{O}(n^3 + n^2 p + r^2 p)$ | $\mathcal{O}(p^3)$ |

update $\boldsymbol{x}$ by $\boldsymbol{x} = (\mu \boldsymbol{I} + \boldsymbol{A})^{-1}\boldsymbol{b}$. If $\boldsymbol{M} = \mu \boldsymbol{I} + 2\gamma \boldsymbol{\Omega}$ then we have

$$\begin{aligned}(\mu \boldsymbol{I} + \boldsymbol{A})^{-1} &= \left(\mu \boldsymbol{I} + 2\gamma \boldsymbol{\Omega} + 2\boldsymbol{X}^T \boldsymbol{X}\right)^{-1} = \left(\boldsymbol{M} + 2\boldsymbol{X}^T \boldsymbol{X}\right)^{-1} \\ &= \boldsymbol{M}^{-1} - 2\boldsymbol{M}^{-1}\boldsymbol{X}^T \left(\boldsymbol{I} + 2\boldsymbol{X}\boldsymbol{M}^{-1}\boldsymbol{X}^T\right)^{-1} \boldsymbol{X}\boldsymbol{M}^{-1}.\end{aligned}$$

The matrix $\boldsymbol{I} + 2\boldsymbol{X}\boldsymbol{M}^{-1}\boldsymbol{X}^T$ is $n \times n$, so we may solve any linear system with this coefficient matrix using $\mathcal{O}(n^3)$ flops; a further $\mathcal{O}(n^2 p)$ flops are needed to compute the coefficient matrix if given $\boldsymbol{M}^{-1}$. Thus, the main computational burden of this update step is the inversion of the matrix $\boldsymbol{M}$. As before, we want to choose $\boldsymbol{\Omega}$ so that we can exploit its structure. Consider the following cases.

- If $\boldsymbol{\Omega} = \mathrm{Diag}(\boldsymbol{u})$ is diagonal, then $\boldsymbol{M}$ is also diagonal with

$$[\boldsymbol{M}^{-1}]_{ii} = \frac{1}{\mu + 2\gamma u_i}.$$

  Thus, we require $\mathcal{O}(p)$ flops to compute $\boldsymbol{M}^{-1}\boldsymbol{v}$ for any vector $\boldsymbol{v} \in \mathbf{R}^p$.
- On the other hand, if $\boldsymbol{\Omega} = \boldsymbol{R}\boldsymbol{R}^T$, where $\boldsymbol{R} \in \mathbf{R}^{p \times r}$, then we may use the Sherman-Morrison-Woodbury identity to compute $\boldsymbol{M}^{-1}$:

$$\boldsymbol{M}^{-1} = \frac{1}{\mu}\boldsymbol{I} - \frac{2\gamma}{\mu^2}\boldsymbol{R}\left(\boldsymbol{I} + \frac{2\gamma}{\mu}\boldsymbol{R}^T\boldsymbol{R}\right)^{-1}\boldsymbol{R}^T.$$

  Therefore, we can solve any linear system with coefficient matrix $\boldsymbol{M}$ at a cost of $\mathcal{O}(r^2 p)$ flops (for the formation and solution of the system with coefficient matrix $\boldsymbol{I} + \frac{2\gamma}{\mu}\boldsymbol{R}^T\boldsymbol{R}$).

In either case, we never actually compute the matrices $\boldsymbol{M}^{-1}$ and $(\mu \boldsymbol{I} + \boldsymbol{A})^{-1}$ explicitly. Instead, we update $\boldsymbol{x}$ as the solution of a sequence of linear systems and matrix–vector multiplications, at a total cost of $\mathcal{O}(n^2 p)$ flops (in the diagonal case) or $\mathcal{O}((r^2 + n^2)p)$ flops (in the factored case). Thus, if the number of observations $n$ is much smaller than the number of features $p$, then the per-iteration computation scales roughly linearly with $p$. Table 5 summarizes these estimates of per-iteration computational costs for each proposed algorithm. Further, we should note that these bounds on per-iteration cost assume that the iterates $\boldsymbol{\beta}$ and $\boldsymbol{x}$ are dense; the soft-thresholding step of the

proximal gradient algorithm typically induces $\boldsymbol{\beta}$ containing many zeros, suggesting that further improvements can be made by using sparse arithmetic.

## Appendix B: Proof of Lemma 1

We note that (3) has trivial solution $\boldsymbol{\theta} = \mathbf{1}_k$ for every $\boldsymbol{\beta} = \boldsymbol{\beta}^i \in \mathbf{R}^p$ and $j = 1$. Indeed, $\boldsymbol{Y}\mathbf{1}_k = \mathbf{1}_n$ by the structure of the indicator matrix $\boldsymbol{Y}$ and $\sum_{i=1}^n x_{ij} = 0$ for all $j = 1, 2, \ldots, p$ because our data has been centered to have sample mean equal to $\mathbf{0}$. Therefore, we may reformulate (3) as

$$\begin{aligned} \min_{\boldsymbol{\theta} \in \mathbf{R}^K} \ & \|\boldsymbol{Y}\boldsymbol{\theta} - \boldsymbol{X}\boldsymbol{\beta}\|^2 \\ \text{s.t. } & \boldsymbol{\theta}^T \boldsymbol{Y}^T \boldsymbol{Y} \boldsymbol{\theta} = n, \\ & \boldsymbol{\theta}^T \boldsymbol{Y}^T \boldsymbol{Y} \mathbf{1} = 0, \\ & \boldsymbol{\theta}^T \boldsymbol{Y}^T \boldsymbol{Y} \boldsymbol{\theta}_\ell = 0 \ \ \ell < j, \end{aligned} \tag{29}$$

to avoid this trivial solution. We wish to show that (29) has optimal solution $\hat{\boldsymbol{\theta}}$ given by

$$\hat{\boldsymbol{\theta}} = \frac{\sqrt{n}\boldsymbol{w}}{\sqrt{\boldsymbol{w}^T \boldsymbol{Y}^T \boldsymbol{Y} \boldsymbol{w}}}, \tag{30}$$

where $\boldsymbol{w} = (\boldsymbol{I} - \frac{1}{n}\boldsymbol{Q}_j \boldsymbol{Q}_j^T \boldsymbol{Y}^T \boldsymbol{Y})(\boldsymbol{Y}^T \boldsymbol{Y})^{-1} \boldsymbol{Y}^T \boldsymbol{X}\boldsymbol{\beta}$.

To do so, note that (29) satisfies the linear independence constraint qualification because the set of constraint function gradients

$$\{2\boldsymbol{Y}^T \boldsymbol{Y}\boldsymbol{\theta}, \boldsymbol{Y}^T \boldsymbol{Y}\mathbf{1}, \boldsymbol{Y}^T \boldsymbol{Y}\boldsymbol{\theta}_1, \ldots, \boldsymbol{Y}^T \boldsymbol{Y}\boldsymbol{\theta}_{j-1}\}$$

is linearly independent. Moreover, the optimal value of (29) is bounded below by 0. Therefore, (29) has global minimizer, $\hat{\boldsymbol{\theta}}$, which must satisfy the Karush-Kuhn-Tucker conditions, i.e., there exists $\boldsymbol{v} \in \mathbf{R}^j$, $\psi \in \mathbf{R}$ such that

$$\boldsymbol{Y}^T \boldsymbol{Y}\hat{\boldsymbol{\theta}} - \boldsymbol{Y}^T \boldsymbol{X}\boldsymbol{\beta} + \psi \boldsymbol{Y}^T \boldsymbol{Y}\hat{\boldsymbol{\theta}} + \boldsymbol{Y}^T \boldsymbol{Y}\boldsymbol{Q}_j \boldsymbol{v} = \mathbf{0}, \tag{31}$$

where $\boldsymbol{Q}_j = [\mathbf{1}, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_{j-1}]$. We consider the following two cases.

First, suppose that $\boldsymbol{Y}^T \boldsymbol{X}\boldsymbol{\beta} \notin \text{range}\left(\boldsymbol{Y}^T \boldsymbol{Y}\boldsymbol{Q}_j\right)$. Rearranging (31) yields

$$\hat{\boldsymbol{\theta}} = \frac{1}{1+\psi}\left(\boldsymbol{Y}^T \boldsymbol{Y}\right)^{-1}\left(\boldsymbol{Y}^T \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{Y}^T \boldsymbol{Y}\boldsymbol{Q}_j \boldsymbol{v}\right). \tag{32}$$

We choose the dual variables $\psi$ and $\boldsymbol{v}$ so that $\hat{\boldsymbol{\theta}}$ is feasible for (29). It is easy to see that the conjugacy constraints are equivalent to $\boldsymbol{Q}_j^T \boldsymbol{Y}^T \boldsymbol{Y}\hat{\boldsymbol{\theta}} = \mathbf{0}$, which holds if and only if

$$\begin{aligned}\mathbf{0} &= \boldsymbol{Q}_j^T(\boldsymbol{Y}^T\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{Q}_j\boldsymbol{v}) = \boldsymbol{Q}_j^T\boldsymbol{Y}^T\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{Q}_j^T\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{Q}_j\boldsymbol{v} \\ &= \boldsymbol{Q}_j^T\boldsymbol{Y}^T\boldsymbol{X}\boldsymbol{\beta} - n\boldsymbol{v},\end{aligned}$$

where the last equality follows from the fact that $\mathbf{1}^T\boldsymbol{Y}^T\boldsymbol{Y}\mathbf{1} = \boldsymbol{\theta}_i^T\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{\theta}_i = n$ for all $i = 1, 2, \ldots, j-1$. It follows immediately that

$$\boldsymbol{v} = \frac{1}{n}\boldsymbol{Q}_j^T\boldsymbol{Y}^T\boldsymbol{X}\boldsymbol{\beta}. \tag{33}$$

Substituting (33) into (32) yields

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \frac{1}{1+\psi}\left((\boldsymbol{Y}^T\boldsymbol{Y})^{-1}\boldsymbol{Y}^T\boldsymbol{X}\boldsymbol{\beta} - \frac{1}{n}\boldsymbol{Q}_j\boldsymbol{Q}_j^T\boldsymbol{Y}^T\boldsymbol{X}\boldsymbol{\beta}\right) \\ &= \frac{1}{1+\psi}\left(\boldsymbol{I} - \frac{1}{n}\boldsymbol{Q}_j\boldsymbol{Q}_j^T\boldsymbol{Y}^T\boldsymbol{Y}\right)(\boldsymbol{Y}^T\boldsymbol{Y})^{-1}\boldsymbol{Y}^T\boldsymbol{X}\boldsymbol{\beta} \\ &= \frac{1}{1+\psi}\boldsymbol{w},\end{aligned} \tag{34}$$

where we choose $\psi \in \mathbf{R}$ so that $\hat{\boldsymbol{\theta}}^T\boldsymbol{Y}^T\boldsymbol{Y}\hat{\boldsymbol{\theta}} = n$:

$$\sqrt{n}(1+\psi) = \pm\sqrt{\boldsymbol{w}^T\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{w}} = \pm\|\boldsymbol{Y}\boldsymbol{w}\|. \tag{35}$$

To complete the argument, note that

$$\begin{aligned}\|\boldsymbol{Y}\hat{\boldsymbol{\theta}} - \boldsymbol{X}\boldsymbol{\beta}\|^2 = n \mp \frac{2\sqrt{n}}{\|\boldsymbol{Y}\boldsymbol{w}\|}\boldsymbol{\beta}^T\boldsymbol{X}^T\left(\boldsymbol{I} - \frac{1}{n}\boldsymbol{Q}_j\boldsymbol{Q}_j^T\boldsymbol{Y}^T\boldsymbol{Y}\right)(\boldsymbol{Y}^T\boldsymbol{Y})^{-1}\boldsymbol{Y}^T\boldsymbol{X}\boldsymbol{\beta} \\ + \|\boldsymbol{X}\boldsymbol{\beta}\|^2.\end{aligned}$$

Note further that the matrix $\boldsymbol{Y}\boldsymbol{Q}_j\boldsymbol{Q}_j^T\boldsymbol{Y}^T$ has decomposition

$$\boldsymbol{Y}\boldsymbol{Q}_j\boldsymbol{Q}_j^T\boldsymbol{Y}^T = \boldsymbol{Y}\mathbf{1}\mathbf{1}^T\boldsymbol{Y}^T + \boldsymbol{Y}\boldsymbol{\theta}_1\boldsymbol{\theta}_1^T\boldsymbol{Y}^T + \cdots + \boldsymbol{Y}\boldsymbol{\theta}_{j-1}\boldsymbol{\theta}_{j-1}^T\boldsymbol{Y}^T.$$

The conjugacy of the columns of $\boldsymbol{Q}_j$ implies that eigenvectors of $\boldsymbol{Y}\boldsymbol{Q}_j\boldsymbol{Q}_j^T\boldsymbol{Y}^T$ are $\boldsymbol{Y}\boldsymbol{\theta}_1, \ldots, \boldsymbol{Y}\boldsymbol{\theta}_{j-1}$, and $\boldsymbol{Y}\mathbf{1}$, each with eigenvalue $n$; since $\boldsymbol{Y}\boldsymbol{Q}_j\boldsymbol{Q}_j^T\boldsymbol{Y}^T$ has rank equal to $k$, all remaining eigenvalues of $\boldsymbol{Y}\boldsymbol{Q}_j\boldsymbol{Q}_j^T\boldsymbol{Y}^T$ must be equal to 0. Moreover, for any $z = \boldsymbol{Y}\boldsymbol{\theta}_i$, $i = 1, 2, \ldots, j-1$, or $z = \boldsymbol{Y}\mathbf{1}$, we have

$$\boldsymbol{Y}\left((\boldsymbol{Y}^T\boldsymbol{Y})^{-1} - \frac{1}{n}\boldsymbol{Q}_j\boldsymbol{Q}_j^T\right)\boldsymbol{Y}^T z = z - z = 0.$$

The matrix $(\boldsymbol{Y}^T\boldsymbol{Y})^{-1}$ is a positive definite diagonal matrix, with $i$th diagonal entry $1/|C_i|$, where $|C_i|$ denotes the number of observations belonging to class $i$; this implies that

$$\boldsymbol{Y}(\boldsymbol{Y}^T\boldsymbol{Y})^{-1}\boldsymbol{Y}^T$$

is positive semidefinite. This establishes that the matrix $\boldsymbol{Y}((\boldsymbol{Y}^T\boldsymbol{Y})^{-1} - \frac{1}{n}\boldsymbol{Q}_j\boldsymbol{Q}_j^T)\boldsymbol{Y}^T$ is positive semidefinite and, thus, $\|\boldsymbol{Y}\boldsymbol{\theta} - \boldsymbol{X}\boldsymbol{\beta}\|^2$ is minimized by $\hat{\boldsymbol{\theta}}$ with $\psi = +\|\boldsymbol{Y}\boldsymbol{w}\|/\sqrt{n} - 1$.

Second, suppose that $\boldsymbol{Y}^T\boldsymbol{X}\boldsymbol{\beta} \in \text{range}\left(\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{Q}_j\right)$. This implies that there exists some $\boldsymbol{v} \in \mathbf{R}^K$ such that

$$\boldsymbol{Y}^T\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{Q}_j\boldsymbol{v}.$$

Substituting into the objective of (29), we see that

$$\begin{aligned}\|\boldsymbol{Y}\boldsymbol{\theta} - \boldsymbol{X}\boldsymbol{\beta}\|^2 &= \boldsymbol{\theta}^T\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{\theta} - 2\boldsymbol{\theta}^T\boldsymbol{Y}^T\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} \\ &= n - 2\boldsymbol{\theta}^T\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{Q}_j\boldsymbol{v} + \boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} \\ &= n + \boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta}\end{aligned}$$

for every feasible solution $\boldsymbol{\theta}$ of (29). This implies that every feasible solution of (29) is also optimal in this case. In particular, $\hat{\boldsymbol{\theta}}$ given by (34) is feasible for (29) and, therefore, optimal. □

## Appendix C: Proof of Theorem 1

We next prove Theorem 1, which establishes that Algorithm 1 converges in function value.

***Proof*** Suppose that, after $t$ iterations, we have iterates $(\boldsymbol{\theta}^i, \boldsymbol{\beta}^i)$ with objective function value $F(\boldsymbol{\theta}^i, \boldsymbol{\beta}^i)$. Recall that we obtain $\boldsymbol{\beta}^{i+1}$ as the solution of (6). Moreover, note that $\boldsymbol{\beta}^i$ is also feasible for (6). This immediately implies that

$$F(\boldsymbol{\theta}^i, \boldsymbol{\beta}^i) \geq F(\boldsymbol{\theta}^i, \boldsymbol{\beta}^{i+1}).$$

On the other hand, $\boldsymbol{\theta}^{i+1}$ is the solution of (3) with $\boldsymbol{\beta} = \boldsymbol{\beta}^{i+1}$. Therefore, we have

$$F(\boldsymbol{\theta}^i, \boldsymbol{\beta}^i) \geq F(\boldsymbol{\theta}^i, \boldsymbol{\beta}^{i+1}) \geq F(\boldsymbol{\theta}^{i+1}, \boldsymbol{\beta}^{i+1}).$$

It follows that the sequence of function values $\{F(\boldsymbol{\theta}^i, \boldsymbol{\beta}^i)\}_{i=1}^{\infty}$ is nonincreasing. Moreover, the objective function $F(\boldsymbol{\theta}, \boldsymbol{\beta})$ is nonnegative for all $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$. Therefore, $\{F(\boldsymbol{\theta}^i, \boldsymbol{\beta}^i)\}_{i=1}^{\infty}$ is convergent as a monotonic bounded sequence. □

## Appendix D: Proof of Theorem 2

To prove Theorem 2, we first establish the following lemma, which establishes that the limit point $(\boldsymbol{\theta}^*, \boldsymbol{\beta}^*)$ minimizes $F$ with respect to each primal variable with the other fixed; that is, $\boldsymbol{\theta}^*$ minimizes $F(\cdot, \boldsymbol{\beta}^*)$ and $\boldsymbol{\beta}^*$ minimizes $F(\boldsymbol{\theta}^*, \cdot)$.

**Lemma 2** *Let* $\{(\boldsymbol{\theta}^i, \boldsymbol{\beta}^i)\}_{i=1}^{\infty}$ *be the sequence of points generated by Algorithm 1. Suppose that* $\{(\boldsymbol{\theta}^{t_j}, \boldsymbol{\beta}^{t_j})\}_{j=1}^{\infty}$ *is a convergent subsequence of* $\{(\boldsymbol{\theta}^i, \boldsymbol{\beta}^i)\}_{i=1}^{\infty}$ *with limit* $(\boldsymbol{\theta}^*, \boldsymbol{\beta}^*)$. *Then*

$$F(\boldsymbol{\theta}, \boldsymbol{\beta}^*) \geq F(\boldsymbol{\theta}^*, \boldsymbol{\beta}^*) \tag{36}$$

$$F(\boldsymbol{\theta}^*, \boldsymbol{\beta}) \geq F(\boldsymbol{\theta}^*, \boldsymbol{\beta}^*) \tag{37}$$

*for all feasible* $\boldsymbol{\theta} \in \mathbf{R}^k$ *and* $\boldsymbol{\beta} \in \mathbf{R}^p$.

***Proof*** We first establish (37). Consider $(\boldsymbol{\theta}^{t_j}, \boldsymbol{\beta}^{t_j})$. By our update step for $\boldsymbol{\beta}$, we note that

$$\boldsymbol{\beta}^{t_j} = \operatorname*{arg\,min}_{\boldsymbol{\beta} \in \mathbf{R}^p} F(\boldsymbol{\theta}^{t_j}, \boldsymbol{\beta}).$$

Thus, for all $j = 1, 2, \ldots$, we have $F(\boldsymbol{\theta}^{t_j}, \boldsymbol{\beta}) \geq F(\boldsymbol{\theta}^{t_j}, \boldsymbol{\beta}^{t_j})$ for all $\boldsymbol{\beta} \in \mathbf{R}^p$. Taking the limit as $j \to \infty$ and using the continuity of $F$ establishes (37).

Next, note that, for every $j = 1, 2, \ldots$, we have

$$\begin{aligned} \boldsymbol{\theta}^{t_j+1} = \operatorname*{arg\,min}_{\boldsymbol{\theta} \in \mathbf{R}^K} \; & F(\boldsymbol{\theta}, \boldsymbol{\beta}^{t_j}) \\ & \text{s.t. } \boldsymbol{\theta}^T \boldsymbol{Y}^T \boldsymbol{Y} \boldsymbol{\theta} = n, \;\; \boldsymbol{\theta}^T \boldsymbol{Y}^T \boldsymbol{Y} \boldsymbol{\theta}_\ell = 0 \;\; \forall \ell < k. \end{aligned}$$

This implies that

$$F(\boldsymbol{\theta}, \boldsymbol{\beta}^{t_j}) \geq F(\boldsymbol{\theta}^{t_j+1}, \boldsymbol{\beta}^{t_j}) \geq F(\boldsymbol{\theta}^{t_j+1}, \boldsymbol{\beta}^{t_j+1}) \geq F(\boldsymbol{\theta}^{t_{j+1}}, \boldsymbol{\beta}^{t_{j+1}})$$

by the monotonicity of the sequence of function values and the fact that $t_j < t_j + 1 \leq t_{j+1}$. Taking the limit as $j \to \infty$ and using the continuity of $F$ establishes (36).

This completes the proof of Lemma 2. □

We are now ready to prove Theorem 2.

***Proof of Theorem 2*** The form of the subdifferential of $\mathcal{L}$ implies that $(\boldsymbol{g}_{\boldsymbol{\theta}}, \boldsymbol{g}_{\boldsymbol{\beta}})$ belongs to the subdifferential $\partial \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\beta}, \psi, \boldsymbol{v})$ if and only if

$$\boldsymbol{g}_{\boldsymbol{\theta}} = 2(1 + \psi)\boldsymbol{Y}^T \boldsymbol{Y} \boldsymbol{\theta} - 2\boldsymbol{Y}^T \boldsymbol{X} \boldsymbol{\beta} + \boldsymbol{U}^T \boldsymbol{v} \tag{38}$$

$$\boldsymbol{g}_{\boldsymbol{\beta}} \in 2(\boldsymbol{X}^T \boldsymbol{X} + \gamma \boldsymbol{\Omega})\boldsymbol{\beta} - 2\boldsymbol{X}^T \boldsymbol{Y} \boldsymbol{\theta} + \lambda \partial \|\boldsymbol{\beta}\|_1 \tag{39}$$

for all $\boldsymbol{v} \in \mathbf{R}^{j-1}$ and $\psi \in \mathbf{R}$.

It is easy to see from (37) that $\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta} \in \mathbf{R}^p} F(\boldsymbol{\theta}^*, \boldsymbol{\beta})$. Thus, by the first order necessary conditions for unconstrained convex optimization, we must have

$$\begin{aligned} \mathbf{0} &\in \partial \left( \frac{1}{2} (\boldsymbol{\beta}^*)^T \boldsymbol{A} \boldsymbol{\beta}^* + \boldsymbol{d}^T \boldsymbol{\beta}^* + \lambda \|\boldsymbol{\beta}^*\|_1 \right) \\ &= 2(\boldsymbol{X}^T \boldsymbol{X} + \gamma \boldsymbol{\Omega})\boldsymbol{\beta}^* - 2\boldsymbol{X}^T \boldsymbol{Y} \boldsymbol{\theta}^* + \lambda \partial \|\boldsymbol{\beta}^*\|_1; \end{aligned} \tag{40}$$

here $\partial\|\boldsymbol{\beta}\|_1$ denotes the subdifferential of the $\ell_1$-norm at the point $\boldsymbol{\beta}$.

On the other hand, (36) implies

$$\begin{aligned}\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}\in\mathbf{R}^K}{\arg\min}\ & \|\boldsymbol{Y\theta} - \boldsymbol{X\beta}^*\|^2 \\ \text{s.t. } & \boldsymbol{\theta}^T\boldsymbol{Y}^T\boldsymbol{Y\theta} = n,\ \ 2\boldsymbol{\theta}^T\boldsymbol{Y}^T\boldsymbol{Y\theta}_\ell = 0,\ \forall\,\ell < j.\end{aligned} \tag{41}$$

Moreover, the problem (41) satisfies the linear independence constraint qualification. Indeed, the set of active constraint gradients $\{2\boldsymbol{Y}^T\boldsymbol{Y\theta}, 2\boldsymbol{Y}^T\boldsymbol{Y\theta}_1, \ldots, 2\boldsymbol{Y}^T\boldsymbol{Y\theta}_{j-1}\}$ is linearly independent for any feasible $\boldsymbol{\theta} \in \mathbf{R}^K$ by the $\boldsymbol{Y}^T\boldsymbol{Y}$-conjugacy of $\{\boldsymbol{\theta}, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{j-1}\}$. Therefore, there exist Lagrange multipliers $\psi^*$, $\boldsymbol{v}^*$ such that

$$\mathbf{0} = 2(1+\psi^*)\boldsymbol{Y}^T\boldsymbol{Y\theta}^* - 2\boldsymbol{Y}^T\boldsymbol{X\beta}^* + \boldsymbol{U}^T\boldsymbol{v}^* \tag{42}$$

by the first-order necessary conditions for optimality (see Nocedal and Wright 2006, Theorem 12.1).

We see that $\mathbf{0} \in \partial\mathcal{L}(\boldsymbol{\theta}^*, \boldsymbol{\beta}^*, \psi^*, \boldsymbol{v}^*)$ by combining (40) and (42). This completes the proof. □

## References


Allen-Zhu Z, Orecchia L (2017) Linear coupling: an ultimate unification of gradient and mirror descent. In: Papadimitriou CH (ed. 8th Innovations in theoretical computer science conference, ITCS 2017, January 9-11, Berkeley, CA, USA, LIPIcs, vol 67, pp 3:1–3:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017). https://doi.org/10.4230/LIPIcs.ITCS.2017.3

Ames B, Hong M (2016) Alternating direction method of multipliers for penalized zero-variance discriminant analysis. Comput Optim Appl 64(3):725–754. https://doi.org/10.1007/s10589-016-9828-y

Beck A (2017) First-order methods in optimization, vol 25. SIAM

Beck A, Teboulle M (2009) A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J Imag Sci 2(1):183–202. https://doi.org/10.1137/080716542

Boyd S, Parikh N, Chu E, Peleato B, Eckstein J (2011) Distributed optimization and statistical learning via the alternating direction method of multipliers. Found Trends Mach Learn 3(1):1–122. https://doi.org/10.1561/2200000016

Bubeck S, Lee YT, Singh M (2015) A geometric alternative to Nesterov's accelerated gradient descent

Cai T, Liu W (2011) A direct estimation approach to sparse linear discriminant analysis. J Am Stat Assoc 106(496):1566–1577. https://doi.org/10.1198/jasa.2011.tm11199

Clemmensen L (2008) Sparse discriminant analysis software (sparseLDA): Matlab and R packages . https://orbit.dtu.dk/en/publications/sparse-discriminant-analysis-software-sparselda-matlab-and-r-pack/. Matlab and R versions of the sparseLDA package

Clemmensen L, Hastie T, Witten D, Ersbøll B (2011) Sparse discriminant analysis. Technometrics 53(4):406–413. https://doi.org/10.1198/TECH.2011.08118

Dau HA, Bagnall AJ, Kamgar K, Yeh CM, Zhu Y, Gharghabi S, Ratanamahatana CA, Keogh EJ (2019) The UCR time series archive. IEEE CAA J Autom Sinica 6(6):1293–1305. https://doi.org/10.1109/jas.2019.1911747

Dau HA, Keogh E, Kamgar K, Yeh CCM, Zhu Y, Gharghabi S, Ratanamahatana CA, Yanping Hu B, Begum N, Bagnall A, Mueen A, Batista G (2018) The UCR time series classification archive. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/

Deng W, Yin W (2012) On the global and linear convergence of the generalized alternating direction method of multipliers. J Sci Comput 3(66):889–916. https://doi.org/10.1007/s10915-015-0048-x

Efron B, Hastie T, Johnstone I, Tibshirani R (2004) Least angle regression. Ann Stat 32(2):407–499

Einarsson G, Clemmensen L, Ames B, Atkins S (2017) Accsda: accelerated sparse discriminant analysis. https://cran.r-project.org/web/packages/accSDA/index.html. Also available at https://github.com/gumeo/accSDA

Einarsson G, Jensen JN, Paulsen RR, Einarsdottir H, Ersbøll BK, Dahl AB, Christensen LB (2017) Foreign object detection in multispectral x-ray images of food items using sparse discriminant analysis. In: Scandinavian conference on image analysis, pp 350–361. Springer

Fan J, Fan Y (2008) High dimensional classification using features annealed independence rules. Ann Stat 36(6):2605–2637. https://doi.org/10.1214/07-AOS504

Flammarion N, Bach F (2015) From averaging to acceleration, there is only a step-size. In: Conference on learning theory, pp 658–695. PMLR

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

Goldfarb D, Ma S, Scheinberg K (2013) Fast alternating linearization methods for minimizing the sum of two convex functions. Math Program 141(1):349–382

Golub GH, Van Loan CF (2013) Matrix computations, 4th edn. The Johns Hopkins University Press, Baltimore

Grosenick L, Greer S, Knutson B (2008) Interpretable classifiers for fmri improve prediction of purchases. IEEE Trans Neural Syst Rehabil Eng 16(6):539–548

Hastie T, Buja A, Tibshirani R (1995) Penalized discriminant analysis. Ann Stat pp 73–102

Hastie T, Tibshirani R, Buja A (1994) Flexible discriminant analysis by optimal scoring. J Am Stat Assoc 89(428):1255–1270. https://doi.org/10.2307/2290989

Hastie T, Tibshirani R, Friedman JH (2013) The elements of statistical learning, 2nd edn. Springer, New York

Hastie T, Tibshirani R, Wainwright M (2012) Statistical learning with sparsity: the lasso and generalizations, 1st edn. CRC Press, Boca Raton

He B, Yuan X (2012) On the $O(1/n)$ convergence rate of the Douglas-Rachford alternating direction method. SIAM J Numer Anal 50(2):700–709

Lessard L, Recht B, Packard A (2016) Analysis and design of optimization algorithms via integral quadratic constraints. SIAM J Optim 26(1):57–95. https://doi.org/10.1137/15M1009597

Ma Q, Yuan M, Zou H (2012) A direct approach to sparse discriminant analysis in ultra-high dimensions. Biometrika 99:29–42. https://doi.org/10.1093/biomet/asr066

Mai Q, Yang Y, Zou H (2019) Multiclass sparse discriminant analysis. Stat Sin 29(1):97–111

Mai Q, Zou H (2013) A note on the connection and equivalence of three sparse linear discriminant analysis methods. Technometrics 55(2):243–246. https://doi.org/10.1080/00401706.2012.746208

Matérn B (2013) Spatial variation, vol 36. Springer Science & Business Media, New York

Merchante LFS, Grandvalet Y, Govaert G (2012) An efficient approach to sparse linear discriminant analysis. In: Proceedings of the 29th international conference on machine learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012. icml.cc / Omnipress. http://icml.cc/2012/papers/591.pdf

Nesterov Y (1983) A method of solving a convex programming problem with convergence rate $O(1/k^2)$. In: Soviet mathematics doklady, vol 27, pp 372–376. http://mpawankumar.info/teaching/cdt-big-data/nesterov83.pdf

Nesterov Y (2005) Smooth minimization of non-smooth functions. Math Program 103(1):127–152. https://doi.org/10.1007/s10107-004-0552-5

Nesterov Y (2013) Gradient methods for minimizing composite functions. Math Program 140(1):125–161. https://doi.org/10.1007/s10107-012-0629-5

Nishihara R, Lessard L, Recht B, Packard A, Jordan M (2015) A general analysis of the convergence of ADMM. In: International conference on machine learning, pp 343–352. PMLR

Nocedal J, Wright S (2006) Numerical optimization, 2nd edn. Springer Science & Business Media, New York

O'Donoghue B, Candes E (2015) Adaptive restart for accelerated gradient schemes. Found Comput Math 15(3):715–732. https://doi.org/10.1007/s10208-013-9150-3

Parikh N, Boyd SP (2014) Proximal algorithms. Found Trends Optim 1(3):127–239. https://doi.org/10.1561/2400000003

Roth V, Fischer B (2008) The group-lasso for generalized linear models: uniqueness of solutions and efficient algorithms. In: Proceedings of the 25th international conference on machine learning, pp 848–855

Shao J, Wang Y, Deng X, Wang S (2011) Sparse linear discriminant analysis by thresholding for high dimensional data. Ann Stat 39(2):1241–1265. https://doi.org/10.1214/10-AOS870

Su W, Boyd S, Candes E (2014) A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights. In: Advances in neural information processing systems, pp 2510–2518

Tibshirani R, Hastie T, Narasimhan B, Chu G (2003) Class prediction by nearest shrunken centroids, with applications to DNA microarrays. Stat Sci. https://doi.org/10.1214/ss/1056397488

Tseng P (2008) On accelerated proximal gradient methods for convex-concave optimization. submitted to SIAM Journal on Optimization 2(3). http://www.mit.edu/~dimitrib/PTseng/papers/apgm.pdf

Witten DM, Tibshirani R (2011) Penalized classification using Fisher's linear discriminant. J R Stat Soc Ser B 73(5):753–772. https://doi.org/10.1111/j.1467-9868.2011.00783.x

Wu M, Zhang L, Wang Z, Christiani D, Lin X (2008) Sparse linear discriminant analysis for simultaneous testing for the significance of a gene set/pathway and gene selection. Bioinformatics 25(9):1145–1151. https://doi.org/10.1093/bioinformatics/btp019

Zou H, Hastie T (2005) Regularization and variable selection via the elastic net. J R Stat Soc Ser B 67(2):301–320. https://doi.org/10.1111/j.1467-9868.2005.00503.x