COMBSS: Best Subset Selection via Continuous Optimization

Sarat Moka (s.moka@unsw.edu.au)
UNSW Sydney
Benoit Liquet
Macquarie University
Houying Zhu
Macquarie University
Samuel Muller
Macquarie University

Research Article

Keywords: Linear regression, High-dimensional regression, Model selection, Variable selection

Posted Date: June 22nd, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3077764/v1

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at Statistics and Computing on February 12th, 2024. See the published version at https://doi.org/10.1007/s11222-024-10387-8.

COMBSS: Best Subset Selection via Continuous Optimization

Sarat Moka[1,2*], Benoit Liquet[2,3], Houying Zhu[2] and Samuel Muller[2,1]

[1*]School of Mathematics and Statistics, The University of New South Wales, Sydney, NSW, Australia.

[2]School of Mathematical and Physical Sciences, Macquarie University, Sydney, NSW, Australia.

[3]Laboratoire de Mathématiques et de leurs Applications, Université de Pau et des Pays de l'Adour, Pau, France.

[4]School of Mathematics and Statistics, The University of Sydney, Sydney, NSW, Australia.

*Corresponding author(s). E-mail(s): s.moka@unsw.edu.au;
Contributing authors: benoit.liquet-weiland@mq.edu.au; houying.zhu@mq.edu.au;
samuel.muller@mq.edu.au;

Abstract

The problem of best subset selection in linear regression is considered with the aim to find a fixed size subset of features that best fits the response. This is particularly challenging when the total available number of features is very large compared to the number of data samples. Existing optimal methods for solving this problem tend to be slow while fast methods tend to have low accuracy. Ideally, new methods perform best subset selection faster than existing optimal methods but with comparable accuracy, or, being more accurate than methods of comparable computational speed. Here, we propose a novel continuous optimization method that identifies a solution path, a small set of models of varying size, that consists of candidates for the best subset in linear regression. Our method turns out to be very fast, making the best subset selection possible when the number of features is well in excess of thousands. Because of the outstanding overall performance, framing the best subset selection challenge as a continuous optimization problem opens new research directions for feature extraction for a large variety of regression models.

Keywords: Linear regression, High-dimensional regression, Model selection, Variable selection

1 Introduction

Recent developments in information technology have enabled the collection of high-dimensional complex data in engineering, economics, finance, biology, health sciences and other fields (Fan and Li, 2006). In high-dimensional data, the number of features is large and often far higher than the number of collected data samples. In many applications, it is desirable to find a parsimonious best subset of predictors so that the resulting model has desirable prediction accuracy (Müller and Welsh, 2010; Fan and Lv, 2010; Miller, 2019). This article is recasting the challenge of best subset selection in linear regression as a novel continuous optimization problem. We show that this reframing has enormous potential and substantially advances research into larger dimensional and exhaustive feature selection in

regression, making available technology that can reliably and exhaustively select variables when the total number of variables is well in excess of thousands.

Here, we aim to develop a method that performs best subset selection and an approach that is faster than existing exhaustive methods while having comparable accuracy, or, that is more accurate than other methods of comparable computational speed.

Consider the linear regression model of the form $\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where $\boldsymbol{y} = (y_1, \ldots, y_n)^\mathsf{T}$ is an n-dimensional known response vector, X is a known design matrix of dimension $n \times p$ with $x_{i,j}$ indicating the ith observation of the jth explanatory variable, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^\mathsf{T}$ is the p-dimensional vector of unknown regression coefficients, and $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_n)^\mathsf{T}$ is a vector of unknown errors, unless otherwise specified, assumed to be independent and identically distributed. Best subset selection is a classical problem that aims to solve,

$$\begin{aligned} &\underset{\boldsymbol{\beta} \in \mathsf{R}^p}{\text{minimize}} \quad \frac{1}{n}\|\boldsymbol{y} - X\boldsymbol{\beta}\|_2^2 \\ &\text{subject to } \|\boldsymbol{\beta}\|_0 = k, \end{aligned} \tag{1}$$

for a given k, where $\|\cdot\|_2$ is the $\mathcal{L}_2$-norm, $\|\boldsymbol{\beta}\|_0 = \sum_{j=1}^p I(\beta_j \neq 0)$ is the number of non-zero elements in $\boldsymbol{\beta}$, and $I(\cdot)$ is the indicator function. For ease of presentation, we assume that all columns of X are subject to selection, but generalizations are immediate (see Remark 2 for more details).

Exact methods for solving (1) are typically executed by first writing solutions for low-dimensional problems and then selecting the best solution over these. To see this, for any binary vector $\boldsymbol{s} = (s_1, \ldots, s_p)^\mathsf{T} \in \{0,1\}^p$, let $X_{[\boldsymbol{s}]}$ be the matrix of size $n \times |\boldsymbol{s}|$ created by keeping only columns j of X for which $s_j = 1$, where $j = 1, \ldots, p$. Then, for any k, in the exact best subset selection, an optimal $\boldsymbol{s}$ can be found by solving the problem,

$$\begin{aligned} &\underset{\boldsymbol{s} \in \{0,1\}^p}{\text{minimize}} \quad \frac{1}{n}\|\boldsymbol{y} - X_{[\boldsymbol{s}]}\widehat{\boldsymbol{\beta}}_{[\boldsymbol{s}]}\|_2^2 \\ &\text{subject to } |\boldsymbol{s}| = k, \end{aligned} \tag{2}$$

where $\widehat{\boldsymbol{\beta}}_{[\boldsymbol{s}]}$ is a low-dimensional least squares estimate of elements of $\boldsymbol{\beta}$ with indices corresponding to non-zero elements of $\boldsymbol{s}$, given by

$$\widehat{\boldsymbol{\beta}}_{[\boldsymbol{s}]} = (X_{[\boldsymbol{s}]}^\mathsf{T} X_{[\boldsymbol{s}]})^\dagger X_{[\boldsymbol{s}]}^\mathsf{T} \boldsymbol{y}, \tag{3}$$

where $A^\dagger$ denotes the pseudo-inverse of a matrix A. Both (1) and (2) are essentially solving the same problem, because $\widehat{\boldsymbol{\beta}}_{[\boldsymbol{s}]}$ is the least squares solution when constrained so that $I(\beta_j \neq 0) = s_j$ for all $j = 1, \ldots, p$.

It is well-known that solving the exact optimization problem above is in general non-deterministic polynomial-time hard (Natarajan, 1995). For instance, a popular exact method called *leaps-and-bounds* (Furnival and Wilson, 2000) is currently practically useful only for values of p smaller than 30 (Tarr et al, 2018). To overcome this difficulty, the relatively recent method by Bertsimas et al (2016) elegantly reformulates the best subset selection problem (1) as a mixed integer optimization and demonstrates that the problem can be solved for p much larger than 30 using modern mixed integer optimization solvers such as in the commercial software Gurobi (Gurobi Optimization, limited liability company, 2022) (which is not freely available except for an initial short period). As the name suggests, the formulation of mixed integer optimization has both continuous and discrete constraints. Although, the mixed integer optimization approach is faster than the exact methods for large p, its implementation via Gurobi remains slow from a practical point of view (Hazimeh and Mazumder, 2020a).

Due to computational constraints of mixed integer optimization, other popular existing methods for best subset selection are still very common in practice, these include forward stepwise selection, the *least absolute shrinkage and selection operator* (generally known as the Lasso), and their variants. Forward stepwise selection follows a greedy approach, starting with an empty model (or intercept-only model), and iteratively adding the variable that is most suitable for inclusion (Efroymson, 1966; Hocking and Leslie, 1967). On the other hand, the Lasso (Tibshirani, 1996) solves a convex relaxation of the highly non-convex best subset selection problem by replacing the discrete $\mathcal{L}_0$-norm $\|\boldsymbol{\beta}\|_0$ in (1) with the $\mathcal{L}_1$-norm $\|\boldsymbol{\beta}\|_1$. This clever relaxation makes the Lasso fast, significantly faster than mixed-integer optimization solvers. However, it is important to note that

Lasso solutions typically do not yield the best subset solution (Hazimeh and Mazumder, 2020a; Zhu et al, 2020) and in essence solve a different problem than exhaustive best subset selection approaches. In summary, there exists a trade-off between speed and accuracy when selecting an existing best subset selection method.

With the aim to develop a method that performs best subset selection as fast as the existing fast methods without compromising the accuracy, in this paper, we design COMBSS, a novel *continuous optimization method towards best subset selection.*

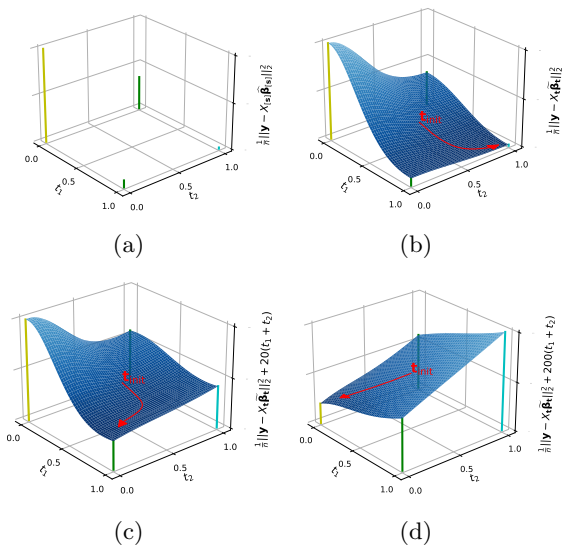


Fig. 1: Illustration of the workings of COMBSS for an example data with $p = 2$. Plot (a) shows the objective function of the exact method (2) for $\boldsymbol{s} \in \{0,1\}^2$. Observe that the best subsets correspond to $k = 0$, $k = 1$, and $k = 2$ are $(1,1)^\top$, $(1,0)^\top$, and $(0,0)^\top$, respectively. Plots (b) - (d) show the objective function of our optimization method (4) for different values of the parameter λ. In each of these three plots, the curve (in red) shows the execution of a basic gradient descent algorithm that, starting at the initial point $\boldsymbol{t}_{init} = (0.5, 0.5)^\top$, converges towards the best subsets of sizes $0, 1$, and 2, respectively.

Our continuous optimization method can be described as follows. Instead of the binary vector space $\{0,1\}^p$ as in the exact methods, we consider the whole hyper-cube $[0,1]^p$ and for each $\boldsymbol{t} \in [0,1]^p$, we consider a new estimate $\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}$ (defined later in Section 2) so that we have the following well-defined continuous extension of the exact problem (2):

$$\underset{\boldsymbol{t}\in[0,1]^p}{\text{minimize}} \quad \frac{1}{n}\|\boldsymbol{y} - X_{\boldsymbol{t}}\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}\|_2^2 + \lambda \sum_{j=1}^{p} t_j, \qquad (4)$$

where $X_{\boldsymbol{t}}$ is obtained from X by multiplying the jth column of X by t_j for each $j = 1, \ldots, p$, and the tuning parameter λ controls the sparsity of the solution obtained, analogous to selecting the best k in the exact optimization. Our construction of $\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}$ guarantees that $\|\boldsymbol{y} - X_{\boldsymbol{s}}\widetilde{\boldsymbol{\beta}}_{\boldsymbol{s}}\|_2 = \|\boldsymbol{y} - X_{[\boldsymbol{s}]}\widehat{\boldsymbol{\beta}}_{[\boldsymbol{s}]}\|_2$ at the corner points $\boldsymbol{s}$ of the hypercube $[0,1]^p$ where $\widehat{\boldsymbol{\beta}}_{[\boldsymbol{s}]}$ exists, and the new objective function $\|\boldsymbol{y} - X_{\boldsymbol{t}}\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}\|_2^2$ is smooth over the hypercube.

While COMBSS aims to find sets of models that are candidates for the best subset of variables, an important property is that it has no discrete constraints, unlike the exact optimization problem (2) or the mixed integer optimization formulation. As a consequence, our method can take advantage of standard continuous optimization methods, such as gradient descent methods, by starting at an interior point on the hypercube $[0,1]^p$ and iteratively moving towards a corner that minimizes the objective function. See Fig. 1 for an illustration of our method. In the implementation, we move the box constrained problem (4) to an equivalent unconstrained problem so that the gradient descent method can run without experiencing boundary issues.

The rest of the paper is organized as follows: In Section 2, we describe the mathematical framework of the proposed method COMBSS. In Section 3, we first establish the continuity of the objective functions involved in COMBSS, and then we derive expressions for their gradients. These gradients are exploited for conducting continuous optimization. Complete details of COMBSS algorithm are presented in Section 4. In Section 5, we discuss roles of the tuning parameters that control the surface shape of the objective functions and the sparsity of the solutions obtained. Section 6 provides steps for efficient implementation of COMBSS using some popular linear algebra techniques. Simulation results

comparing COMBSS with existing popular methods are presented in Section 7. We conclude the paper with some brief remarks in Section 8. Proofs all our theoretical results are provided in Appendix A.

2 Continuous Extension of Best Subset Selection Problem

To see our continuous extension of the exact best subset selection optimization problem (2), for $\boldsymbol{t} = (t_1, \ldots, t_p)^\top \in [0,1]^p$, define $T_{\boldsymbol{t}} = \mathsf{Diag}(\boldsymbol{t})$, the diagonal matrix with the diagonal elements being $t_1, \ldots, t_p$, and let $X_{\boldsymbol{t}} = XT_{\boldsymbol{t}}$. For a fixed constant $\delta > 0$, define

$$L_{\boldsymbol{t}} = L_{\boldsymbol{t}}(\delta) = \frac{1}{n}\left[X_{\boldsymbol{t}}^\top X_{\boldsymbol{t}} + \delta\left(I - T_{\boldsymbol{t}}^2\right)\right], \tag{5}$$

where we suppress δ for ease of reading. Intuitively, $L_{\boldsymbol{t}}$ can be seen as a 'convex combination' of the matrices $(X^\top X)/n$ and $\delta I/n$, because $X_{\boldsymbol{t}}^\top X_{\boldsymbol{t}} = T_{\boldsymbol{t}} X^\top X T_{\boldsymbol{t}}$ and thus

$$L_{\boldsymbol{t}} = T_{\boldsymbol{t}} \frac{X^\top X}{n} T_{\boldsymbol{t}} + (I - T_{\boldsymbol{t}})\,\frac{\delta\, I}{n}\,(I - T_{\boldsymbol{t}}). \tag{6}$$

Using this notation, now define

$$\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}} = L_{\boldsymbol{t}}^\dagger \left(\frac{X_{\boldsymbol{t}}^\top \boldsymbol{y}}{n}\right), \quad \boldsymbol{t} \in [0,1]^p. \tag{7}$$

We need $L_{\boldsymbol{t}}^\dagger$ in (7) so that $\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}$ is defined for all $\boldsymbol{t} \in [0,1]^p$. However, from the way we conduct optimization, we need to compute $\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}$ only for $\boldsymbol{t} \in [0,1)^p$. We later show in Theorem 1 that for all $\boldsymbol{t} \in [0,1)^p$, $L_{\boldsymbol{t}}$ is invertible and thus in the implementation of our method, $\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}$ always takes the form $\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}} = L_{\boldsymbol{t}}^{-1} X_{\boldsymbol{t}}^\top \boldsymbol{y}/n$, eliminating the need to compute any computationally expensive pseudo-inverse.

With the support of these observations, an immediate well-defined generalization of the best subset selection problem (1) is

$$\begin{aligned} &\underset{\boldsymbol{t}\in[0,1]^p}{\text{minimize}} \quad \frac{1}{n}\|\boldsymbol{y} - X_{\boldsymbol{t}}\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}\|_2^2 \\ &\text{subject to} \sum_{j=1}^p t_j = k. \end{aligned} \tag{8}$$

Instead of solving the constrained problem (8), by defining a Lagrangian function

$$f_\lambda(\boldsymbol{t}) = \frac{1}{n}\|\boldsymbol{y} - X_{\boldsymbol{t}}\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}\|_2^2 + \lambda \sum_{j=1}^p t_j, \tag{9}$$

for a tunable parameter $\lambda > 0$, we aim to solve

$$\underset{\boldsymbol{t}\in[0,1]^p}{\text{minimize}}\ f_\lambda(\boldsymbol{t}). \tag{10}$$

By defining $g_\lambda(\boldsymbol{w}) = f_\lambda(\boldsymbol{t}(\boldsymbol{w}))$, we reformulate the box constrained problem (10) into an equivalent unconstrained problem,

$$\underset{\boldsymbol{w}\in\mathsf{R}^p}{\text{minimize}}\ g_\lambda(\boldsymbol{w}), \tag{11}$$

where the mapping $\boldsymbol{t} = \boldsymbol{t}(\boldsymbol{w})$ is

$$t_j(w_j) = 1 - \exp(-w_j^2), \quad j = 1, \ldots, p. \tag{12}$$

The unconstrained optimization problem (11) is equivalent to the box constrained problem (10), because $1 - \exp(-u^2) < 1 - \exp(-v^2)$ if and only if $u^2 < v^2$, for any $u, v \in \mathbb{R}$.

Remark 1 The non-zero parameter δ is important in the expression of the proposed estimator $\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}$, as in (7), not only to make $L_{\boldsymbol{t}}$ invertible for $\boldsymbol{t} \in [0,1)^p$, but also to make the surface of $f_\lambda(\boldsymbol{t})$ to have smooth transitions from one corner to another over the hypercube. For example consider a situation where $X^\top X$ is invertible. Then, for any interior point $\boldsymbol{t} \in (0,1)^p$, since $T_{\boldsymbol{t}}^{-1}$ exists, the optimal solution to $\min_{\boldsymbol{\beta}} \|y - X_{\boldsymbol{t}}\boldsymbol{\beta}\|_2^2/n$ after some simplification is $T_{\boldsymbol{t}}^{-1}(X^\top X)X^\top y$. As a result, the corresponding minimum loss is $\|y - X(X^\top X)X^\top y\|_2^2/n$, which is a constant for all $\boldsymbol{t}$ over the interior of the hypercube. Hence, the surface of the loss function would have jumps at the borders while being flat over the interior of the hypercube. Clearly, such a loss function is not useful for conducting continuous optimization.

Remark 2 The proposed method and the corresponding theoretical results presented in this paper easily extend to linear models with intercept term. More generally, if we want to keep some features in the model, say features $j = 1, 2$, and 4, then we enforce $t_j = 1$ for $j = 1, 2, 4$, and conduct subset selection only over the remaining features by taking $\boldsymbol{t} = (1, 1, t_3, 1, t_5, \ldots, t_p)^\top$ and optimize $t_3, t_5, \ldots, t_p$.

3 Continuity and Gradients of the Objective Function

In this section, we first prove that the objective function $g_\lambda(\boldsymbol{w})$ of the unconstrained optimization problem (11) is continuous on $\mathbb{R}^p$ and then we derive its gradients. En-route, we also establish the relationship between $\widehat{\boldsymbol{\beta}}_{[\boldsymbol{s}]}$ and $\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}$ which are respectively defined by (3) and (7). This relationship is useful in understanding the relationship between our method and the exact optimization (2).

Theorem 1 shows that for all $\boldsymbol{t} \in [0,1)^p$, the matrix $L_{\boldsymbol{t}}$, which is defined in (5), is symmetric positive-definite and hence invertible.

Theorem 1 *For any $\boldsymbol{t} \in [0,1)^p$, $L_{\boldsymbol{t}}$ is symmetric positive-definite and $\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}} = L_{\boldsymbol{t}}^{-1} X_{\boldsymbol{t}}^{\mathsf{T}} \boldsymbol{y}/n$.*

Recall that $\widehat{\boldsymbol{\beta}}_{[\boldsymbol{s}]}$ exists only on some corner points $\boldsymbol{s}$ of the hyper-cube $[0,1]^p$, where the matrix $X_{[\boldsymbol{s}]}^{\mathsf{T}} X_{[\boldsymbol{s}]}$ is invertible. Theorem 2 establishes a relationship between $\widetilde{\boldsymbol{\beta}}_{\boldsymbol{s}}$ and $\widehat{\boldsymbol{\beta}}_{[\boldsymbol{s}]}$ at these corner points $\boldsymbol{s} \in \{0,1\}^p$. Towards this, for any point $\boldsymbol{s} \in \{0,1\}^p$ and a vector $\boldsymbol{u} \in \mathbb{R}^p$, we write $(\boldsymbol{u})_+$ (respectively, $(\boldsymbol{u})_0$) to denote the *sliced* vector of dimension $|\boldsymbol{s}|$ (respectively, $p - |\boldsymbol{s}|$) created from $\boldsymbol{u}$ by removing all its elements with the indices j with $s_j = 0$ (respectively, with $s_j > 0$). For instance, if $\boldsymbol{u} = (2,3,4,5)^{\mathsf{T}}$ and $\boldsymbol{s} = (1,0,1,0)^{\mathsf{T}}$, then $(\boldsymbol{u})_+ = (2,4)$ and $(\boldsymbol{u})_0 = (3,5)$.

Theorem 2 *For any $\boldsymbol{s} \in \{0,1\}^p$, $(\widetilde{\boldsymbol{\beta}}_{\boldsymbol{s}})_+ = \widehat{\boldsymbol{\beta}}_{[\boldsymbol{s}]}$ and $(\widetilde{\boldsymbol{\beta}}_{[\boldsymbol{s}]})_0 = \boldsymbol{0}$. Furthermore, we have $X_{[\boldsymbol{s}]}\widehat{\boldsymbol{\beta}}_{[\boldsymbol{s}]} = X_{\boldsymbol{s}}\widetilde{\boldsymbol{\beta}}_{\boldsymbol{s}}$.*

As an immediate consequence of Theorem 2, we have $\|\boldsymbol{y} - X_{[\boldsymbol{s}]}\widehat{\boldsymbol{\beta}}_{[\boldsymbol{s}]}\|_2^2 = \|\boldsymbol{y} - X_{\boldsymbol{s}}\widetilde{\boldsymbol{\beta}}_{\boldsymbol{s}}\|_2^2$. Therefore, the objective function of the exact optimization problem (2) is identical to the objective function of its extended optimization problem (8) (with $\lambda = 0$) at the corner points $\boldsymbol{s} \in \{0,1\}^p$.

Our next result, Theorem 3, shows that $f_\lambda(\boldsymbol{t})$ is a continuous function on $[0,1]^p$.

Theorem 3 *The function $f_\lambda(\boldsymbol{t})$ defined in (9) is continuous over $[0,1]^p$ in the sense that for any sequence $\boldsymbol{t}^{(1)}, \boldsymbol{t}^{(2)}, \cdots \in [0,1)^p$ converging to $\boldsymbol{t} \in [0,1]^p$, the limit $\lim_{l\to\infty} f_\lambda(\boldsymbol{t}^{(l)})$ exists and*

$$f_\lambda(\boldsymbol{t}) = \lim_{l\to\infty} f_\lambda(\boldsymbol{t}^{(l)}).$$

Corollary 1 establishes the continuity of g_λ on $\mathbb{R}^p$. This is a simple consequence of Theorem 3, because from the definition, $g_\lambda(\boldsymbol{w}) = f_\lambda(\boldsymbol{t}(\boldsymbol{w}))$ with $\boldsymbol{t}(\boldsymbol{w}) = \boldsymbol{1} - \exp(-\boldsymbol{w} \odot \boldsymbol{w})$. Here and afterwards, in an expression with vectors, $\boldsymbol{1}$ denotes a vector of all ones of appropriate dimension, $\odot$ denotes the element-wise product of two vectors, and the exponential function, $\exp(\cdot)$, is also applied element-wise.

Corollary 1 The objective function $g_\lambda(\boldsymbol{w})$ is continuous at every point $\boldsymbol{w} \in \mathbb{R}^p$.

As mentioned earlier, our continuous optimization method uses a gradient descent method to solve the problem (11). Towards that we need to obtain the gradients of $g_\lambda(\boldsymbol{w})$. Theorem 4 provides an expression of the gradient $\nabla g_\lambda(\boldsymbol{w})$.

Theorem 4 *For every $\boldsymbol{w} \in \mathbb{R}^p$, with $\boldsymbol{t} = \boldsymbol{t}(\boldsymbol{w})$ is defined by (12),*

$$\nabla f_\lambda(\boldsymbol{t}) = \boldsymbol{\zeta}_{\boldsymbol{t}} + \lambda \boldsymbol{1}, \quad \boldsymbol{t} \in (0,1)^p,$$

and

$$\nabla g_\lambda(\boldsymbol{w}) = (\boldsymbol{\zeta}_{\boldsymbol{t}} + \lambda \boldsymbol{1}) \odot (2\boldsymbol{w} \odot \exp(-\boldsymbol{w} \odot \boldsymbol{w})), \quad \boldsymbol{w} \in \mathbb{R}^p,$$

where

$$\boldsymbol{\zeta}_{\boldsymbol{t}} = 2\left(\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}} \odot (\boldsymbol{a}_{\boldsymbol{t}} - \boldsymbol{d}_{\boldsymbol{t}})\right) - 2\left(\boldsymbol{b}_{\boldsymbol{t}} \odot \boldsymbol{c}_{\boldsymbol{t}}\right), \tag{13}$$

with

$$\begin{aligned}
\boldsymbol{a}_{\boldsymbol{t}} &= \frac{1}{n}[X^{\mathsf{T}} X(\boldsymbol{t} \odot \widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}) - X^{\mathsf{T}} \boldsymbol{y}], \\
\boldsymbol{b}_{\boldsymbol{t}} &= \boldsymbol{a}_{\boldsymbol{t}} - n^{-1}\delta(\boldsymbol{t} \odot \widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}), \\
\boldsymbol{c}_{\boldsymbol{t}} &= L_{\boldsymbol{t}}^{-1}\left(\boldsymbol{t} \odot \boldsymbol{a}_{\boldsymbol{t}}\right), \quad \textit{and} \\
\boldsymbol{d}_{\boldsymbol{t}} &= \frac{1}{n}[X^{\mathsf{T}} X - \delta I](\boldsymbol{t} \odot \boldsymbol{c}_{\boldsymbol{t}}).
\end{aligned}$$

Figure 2 illustrates the typical convergence behavior of $\boldsymbol{t}$ for an example dataset during the execution of a basic gradient descent algorithm for minimizing $g_\lambda(\boldsymbol{w})$ using the gradient ∇g_λ given in Theorem 4. Here, $\boldsymbol{w}$ is mapped to $\boldsymbol{t}$ via (12) at each iteration.

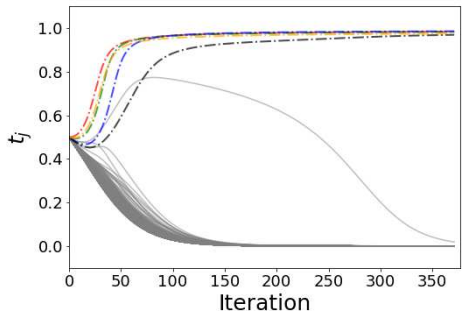


Fig. 2: Convergence of $\boldsymbol{t}$ for a dataset with $p = 1000$ and $n = 100$ during the execution of basic gradient descent. Solid lines correspond to $\beta_j = 0$ and remaining 5 curves (with line style $-\cdot-$) correspond to $\beta_j \neq 0$. The dataset is generated using Beta-type 1 model presented in Section 7 of preprint of this work (Moka et al, 2022) with $k_0 = 5$, $n = 100$, $p = 1000$, $\rho = 0.8$, and signal-to-noise ratio of 5. The parameters $\lambda = 0.1$ and $\delta = n$; see Section 5 for more discussion on how to choose λ and δ.

4 Subset Selection Algorithms

Our algorithm COMBSS as stated in Algorithm 1, takes the data $[X, \boldsymbol{y}]$, tuning parameters δ, λ, and an initial point $\boldsymbol{w}^{(0)}$ as input, and returns either a single model or multiple models of different sizes as output. It is executed in three steps.

In Step 1, $\mathsf{GradientDescent}\left(\boldsymbol{w}^{(0)}, \nabla g_\lambda\right)$ calls a gradient descent method, such as the well known *adam* optimizer, for minimizing the objective function $g_\lambda(\boldsymbol{w})$, which takes $\boldsymbol{w}^{(0)}$ as the initial point and uses the gradient function ∇g_λ for updating the vector $\boldsymbol{w}$ in each iteration; see, for example, Kochenderfer and Wheeler (2019) for a review of popular gradient based optimization methods. It terminates when a predefined termination condition is satisfied and returns the sequence $\boldsymbol{w}_{\mathsf{path}} = (\boldsymbol{w}^{(0)}, \boldsymbol{w}^{(1)}, \dots)$ of all the points $\boldsymbol{w}$ visited during its execution, where $\boldsymbol{w}^{(l)}$ denotes the point obtained in the lth iteration. Usually, a robust termination condition is to terminate when the change in $\boldsymbol{w}$ (or, equivalently, in $\boldsymbol{t}(\boldsymbol{w})$) is significantly small over a set of consecutive iterations.

Selecting the initial point $\boldsymbol{w}^{(0)}$ requires few considerations. From Theorem 4, for any $j = 1, \dots, p$, we have $t_j(w_j) = 0$ if and only if $w_j = 0$ and $\partial g_\lambda(\boldsymbol{w})/\partial w_j = 0$ if $w_j = 0$. Hence,

Algorithm 1 $\mathsf{COMBSS}\left(X, \boldsymbol{y}, \delta, \lambda, \boldsymbol{w}^{(0)}\right)$

1: $\boldsymbol{w}_{\mathsf{path}} \leftarrow \mathsf{GradientDescent}\left(\boldsymbol{w}^{(0)}, \nabla g_\lambda\right)$
2: Obtain $\boldsymbol{t}_{\mathsf{path}}$ from $\boldsymbol{w}_{\mathsf{path}}$ using the map (12)
3: $\mathcal{M} \leftarrow \mathsf{SubsetMap}(\boldsymbol{t}_{\mathsf{path}})$
4: **return** $\mathcal{M}$

if we start the gradient descent algorithm with $w_j^{(0)} = 0$ for some j, both w_j and t_j can continue to take 0 forever. As a result, we might not learn the optimal value for w_j (or, equivalently for t_j). Thus, it is important to select all the elements of $w^{(0)}$ away from 0.

Consider the second argument, ∇g_λ, in the gradient descent method. From Theorem 4, observe that computing the gradient $\nabla g_\lambda(\boldsymbol{w})$ involves finding the values of the expression of the form $L_{\boldsymbol{t}}^{-1}\boldsymbol{u}$ twice, first for computing $\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}$ (using (7)) and then for computing the vector $\boldsymbol{c}_{\boldsymbol{t}}$ (defined in Theorem 4). Since $L_{\boldsymbol{t}}$ is of dimension $p \times p$, computing the matrix inversion $L_{\boldsymbol{t}}^{-1}$ can be computationally demanding particularly in high-dimensional cases ($n < p$), where p can be very large; see, for example, Golub and Van Loan (1996). Since $L_{\boldsymbol{t}}$ is invertible, observe that $L_{\boldsymbol{t}}^{-1}\boldsymbol{u}$ is the unique solution of the linear equation $L_{\boldsymbol{t}}\boldsymbol{z} = \boldsymbol{u}$. In Section 6, we first use the well-known Woodbury matrix identity to convert this p-dimensional linear equation problem to an n-dimensional linear equation problem, which is then solved using the conjugate gradient method, a popular linear equation solver. Moreover, again from Theorem 4, notice that $\nabla g_\lambda(\boldsymbol{w})$ depends on both the tuning parameters δ and λ. Specifically, δ is required for computing $L_{\boldsymbol{t}}$ and λ is used in the penalty term $\lambda \sum_{j=1}^{p} t_j$ of the objective function. In Section 5 we provide more details on the roles of these two parameters and instructions on how to choose them.

In Step 2, we obtain the sequence $\boldsymbol{t}_{\mathsf{path}} = (\boldsymbol{t}^{(0)}, \boldsymbol{t}^{(1)}, \dots)$ from $\boldsymbol{w}_{\mathsf{path}}$ by using the map (12), that is, $\boldsymbol{t}^{(l)} = \boldsymbol{t}(\boldsymbol{w}^{(l)}) = \mathbf{1} - \exp(-\boldsymbol{w}^{(l)} \odot \boldsymbol{w}^{(l)})$ for each l.

Finally, in Step 3, $\mathsf{SubsetMap}(\boldsymbol{t}_{\mathsf{path}})$ takes the sequence $\boldsymbol{t}_{\mathsf{path}}$ as input to find a set of models $\mathcal{M}$ correspond to the input parameter λ. In the following subsections, we describe two versions of $\mathsf{SubsetMap}$.

4.1 Subset Map Version 1

One simple implementation of SubsetMap is stated as Algorithm 2 which we call SubsetMapV1 (where V1 stands for *version 1*) and it requires only the final point in the sequence $\boldsymbol{t}_{\mathsf{path}}$ and returns only one model using a predefined threshold parameter $\tau \in [0, 1)$.

Algorithm 2 SubsetMapV1 $(\boldsymbol{t}_{\mathsf{path}}, \tau)$

Take $\boldsymbol{t}$ to be the final point of $\boldsymbol{t}_{\mathsf{path}}$
for $j = 1$ to $j = p$ **do**
    $s_j \leftarrow \mathbb{I}(t_j > \tau)$
end for
return $\boldsymbol{s} = (s_1, \ldots, s_p)^{\mathsf{T}}$

Due to the tolerance allowed by the termination condition of the gradient descent, some w_j in the final point of $\boldsymbol{w}_{\mathsf{path}}$ can be almost zero but not exactly zero, even though they are meant to converge to zero. As a result, the corresponding t_j also take values close to zero but not exactly zero because of the mapping from $\boldsymbol{w}$ to $\boldsymbol{t}$. Therefore, the threshold τ helps in mapping the insignificantly small t_j to 0 and all other t_j to 1. In practice, we call ModelSelection $\left(X, \boldsymbol{y}, \delta, \lambda, \boldsymbol{w}^{(0)}\right)$ for each λ over a grid of values. When SubsetMapV1 is used, larger the value of λ, higher the sparsity in the resulting model $\boldsymbol{s}$. Thus, we can control the sparsity of the output model using λ. Since we only care about the last point in $\boldsymbol{t}_{\mathsf{path}}$ in this version, an intuitive option for initialization is to take $\boldsymbol{w}^{(0)}$ to be such that $\boldsymbol{t}(\boldsymbol{w}^{(0)}) = (1/2, \ldots, 1/2)^{\mathsf{T}}$, the mid-point on the hypercube $[0, 1]^p$, as it is at an equal distance from all the corner points, of which one is the (unknown) target solution of the best subset selection problem.

In the preprint Moka et al (2022), we demonstrated the efficacy of COMBSS using SubsetMapV1 in predicting the true model of the data. In almost all the cases, we observe superior performance of COMBSS in comparison to existing popular methods.

4.2 Subset Map Version 2

Ideally, there is a value of λ for each k such that the output model $\boldsymbol{s}$ obtained by SubsetMapV1 has exactly k non-zero elements. However, when the ultimate goal is to find a best suitable model $\boldsymbol{s}$ for a given $k \leq q$ such that $|\boldsymbol{s}| = k$, for some $q \ll min(n, p)$, since λ is selected over a grid, we might not obtain any model for some values of k. Furthermore, for a given size k, if there are two models with almost the same mean square error, then the optimization may have difficulty in distinguishing them. Addressing this difficulty may involve fine tuning of hyper-parameters of the optimization algorithm.

To overcome these challenges without any hyper-parameter tuning and reduce the reliance on the parameter λ, we consider the other points in $\boldsymbol{t}_{\mathsf{path}}$. In particular, we propose a more optimal implementation of SubsetMap, which we call SubsetMapV2 and is stated as Algorithm 3. The key idea of this version is that as the gradient descent progresses over the surface of $f_\lambda(\boldsymbol{t})$, it can point towards some corners of the hypercube $[0, 1]^p$ before finally moving towards the final corner. Considering all these corners, we can refine the results. Specifically, this version provides for each λ a model for every $k \leq q$. In this implementation, λ is seen as a parameter that allows us to explore the surface of $f_\lambda(\boldsymbol{t})$ rather than as a sparsity parameter.

Algorithm 3 SubsetMapV2 $(\boldsymbol{t}_{\mathsf{path}})$

$\mathcal{M}_k \leftarrow \{\}$ for each $k \leq q$
for each $\boldsymbol{t} = (t_1, \ldots, t_p)^{\mathsf{T}}$ in $\boldsymbol{t}_{\mathsf{path}}$ **do**
    Let $t_{j_1}, t_{j_2}, \ldots, t_{j_q}$ be the q largest elements of $\boldsymbol{t}$ in the descending order
    for $k = 1$ to q **do**
        Take $\boldsymbol{s}_k \in \{0, 1\}^p$ with non-zero elements only at $j_1, \ldots, j_k$
        $\mathcal{M}_k \leftarrow \mathcal{M}_k \cup \{\boldsymbol{s}_k\}$
    end for
end for
for $k = 1$ to $k = q$ **do**
    $\boldsymbol{s}_k^* \leftarrow \text{argmin}_{\boldsymbol{s} \in \mathcal{M}_k} \ \frac{1}{n}\|\boldsymbol{y} - X_{[\boldsymbol{s}]}\widehat{\boldsymbol{\beta}}_{[\boldsymbol{s}]}\|_2^2$
end for
return $\mathcal{M} = \{\boldsymbol{s}_1^*, \ldots, \boldsymbol{s}_q^*\}$

For the execution of SubsetMapV2, we start at Step 1 with an empty set of models $\mathcal{M}_k$ for each $k \leq q$. In Step 2, for each $\boldsymbol{t}$ in $\boldsymbol{t}_{\mathsf{path}}$, we consider the sequence of indices $j_1, \ldots, j_q$ such that $t_{j_1} \geq t_{j_2} \geq \cdots \geq t_{j_q}$. Then, for each $k \leq q$, we take $\boldsymbol{s}_k$ to be a binary vector with 1's only at $j_1, \ldots, j_k$

and add $\boldsymbol{s}_k$ to the set $\mathcal{M}_k$. With this construction, it is clear that $\mathcal{M}_k$ consists of models of size k, of which we pick a best candidate $\boldsymbol{s}_k^*$ as show at Step 3. Finally, the algorithm returns the set consists of $\boldsymbol{s}_1^*, \ldots, \boldsymbol{s}_q^*$ correspond to the given λ. When the main COMBSS is called for a grid of m values of λ with SubsetMapV2, then for each $k \leq q$ we obtain at most m models and among them the model with the minimum mean squared error is selected as the final best model for k. Since this version of COMBSS explores the surface, we can refine results further by starting from different initial points $\boldsymbol{w}^{(0)}$.

Remark 3 It is not hard to observe that for each λ, if the model obtained by Algorithm 2 is of a size $k \leq q$, then this model is present in $\mathcal{M}_k$ of Algorithm 3, and hence, COMBSS with SubsetMapV2 always provides the same or a better solution than COMBSS with SubsetMapV1.

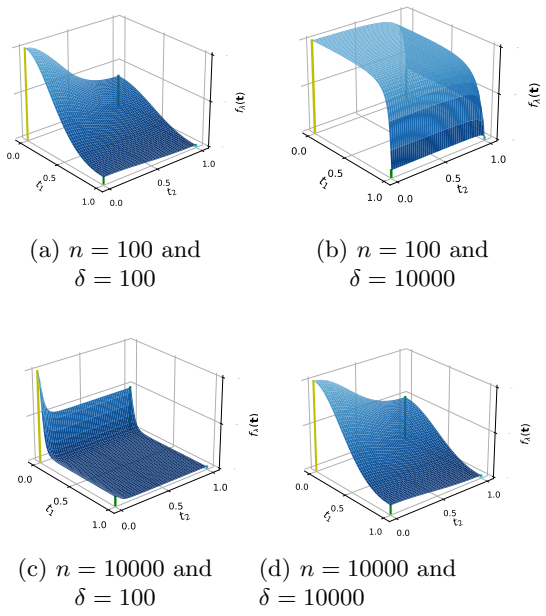
5 Roles of Tuning Parameters

In this section, we provide insights on how the tuning parameters δ and λ influence the objective function $f_\lambda(\boldsymbol{t})$ (or, equivalently $g_\lambda(\boldsymbol{w})$) and hence the convergence of the algorithm.

5.1 Controlling the Shape of $f_\lambda(\boldsymbol{t})$ through δ

The normalized cost $\|\boldsymbol{y} - X_{\boldsymbol{t}}\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}\|_2^2/n$ provides an estimator of the error variance. For any fixed $\boldsymbol{t}$, we expect this variance (and hence the objective function $f_\lambda(\boldsymbol{t})$) to be almost the same for all relatively large values of n, particularly, in situations where the errors ϵ_i are independent and identically distributed. This is the case at all the corner points $\boldsymbol{s} \in \{0,1\}^p$, because at these corner points, from Theorem 2, $X_{\boldsymbol{s}}\widetilde{\boldsymbol{\beta}}_{\boldsymbol{s}} = X_{[\boldsymbol{s}]}\widehat{\boldsymbol{\beta}}_{[\boldsymbol{s}]}$, which is independent of δ. We would like to have a similar behavior at all the interior points $\boldsymbol{t} \in (0,1)^p$ as well, so that for each $\boldsymbol{t}$, the function $f_\lambda(\boldsymbol{t})$ is roughly the same for all large values of n. Such consistent behavior is helpful in guaranteeing that the convergence paths of the gradient descent method are approximately the same for large values of n.

Figure 3 shows surface plots of $f_\lambda(\boldsymbol{t})$ for different values of n and δ for an example dataset



(a) $n = 100$ and $\delta = 100$

(b) $n = 100$ and $\delta = 10000$

(c) $n = 10000$ and $\delta = 100$

(d) $n = 10000$ and $\delta = 10000$

Fig. 3: Illustration of how δ effects the objective function $f_\lambda(\boldsymbol{t})$ (with $\lambda = 0$). A dataset consists of 10000 samples generated from the illustrative linear model used in Figure 1. For (a) and (b), 100 samples from the same dataset are used.

obtained from a linear model with $p = 2$. Surface plots (a) and (d) correspond to $\delta = n$, and as we can see, the shape of the surface of $f_\lambda(\boldsymbol{t})$ over $[0,1]^p$ is very similar in both these plots.

To make this observation more explicit, we now show that the function $f_\lambda(\boldsymbol{t})$, at any $\boldsymbol{t}$, takes almost the same value for all large n if we keep $\delta = c\,n$, for a fixed constant $c > 0$, under the assumption that the data samples are independent and identically distributed (this assumption simplifies the following discussion; however, the conclusion holds more generally).

Observe that

$$\frac{1}{n}\|\boldsymbol{y} - X_{\boldsymbol{t}}\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}\|_2^2 = \frac{\boldsymbol{y}^{\mathsf{T}}\boldsymbol{y}}{n} - 2\,\boldsymbol{\gamma}_{\boldsymbol{t}}^{\mathsf{T}}\frac{X^{\mathsf{T}}\boldsymbol{y}}{n} + \boldsymbol{\gamma}_{\boldsymbol{t}}^{\mathsf{T}}\frac{X^{\mathsf{T}}X}{n}\boldsymbol{\gamma}_{\boldsymbol{t}},$$

where $\boldsymbol{\gamma}_{\boldsymbol{t}} = n^{-1}\,T_{\boldsymbol{t}}L_{\boldsymbol{t}}^{-1}T_{\boldsymbol{t}}X^{\mathsf{T}}\boldsymbol{y}$. Since, under the independent and identically distributed assumption, $\boldsymbol{y}^{\mathsf{T}}\boldsymbol{y}/n$, $X^{\mathsf{T}}\boldsymbol{y}/n$, and $X^{\mathsf{T}}X/n$ converge element-wise as n increases and $T_{\boldsymbol{t}}$ is independent of n, we would like to choose δ such that $L_{\boldsymbol{t}}^{-1}$ also

converges as n increases. Now recall from (6) that

$$L_{\boldsymbol{t}} = T_{\boldsymbol{t}} \left(\frac{X^{\mathsf{T}} X}{n} \right) T_{\boldsymbol{t}} + \frac{\delta}{n} \left(I - T_{\boldsymbol{t}}^2 \right).$$

It is then evident that the choice $\delta = c\, n$ for a fixed constant c, independent of n, makes $L_{\boldsymbol{t}}$ converging as n increases. Specifically, the choice $c = 1$ justifies the behavior observed in Figure 3.

5.2 Sparsity Controlling through λ

Intuitively, the larger the value of λ the sparser the solution offered by COMBSS using SubsetMapV1, when all other parameters are fixed. We now strengthen this understanding mathematically. From Theorem 4,

$$\nabla f_\lambda(\boldsymbol{t}) = \boldsymbol{\zeta}_{\boldsymbol{t}} + \lambda \mathbf{1}, \quad \boldsymbol{t} \in (0,1)^p,$$

and

$$\nabla g_\lambda(\boldsymbol{w}) = (\boldsymbol{\zeta}_{\boldsymbol{t}} + \lambda \mathbf{1}) \odot (2\boldsymbol{w} \odot \exp(-\boldsymbol{w} \odot \boldsymbol{w})),$$

for $\boldsymbol{w} \in \mathbb{R}^p$, where $\boldsymbol{\zeta}_{\boldsymbol{t}}$, given by (13), is independent of λ. Note the following property of $\boldsymbol{\zeta}_{\boldsymbol{t}}$.

Proposition 5 *For any $j = 1, \ldots, p$, if all t_i for $i \neq j$ are fixed,*

$$\lim_{t_j \downarrow 0} \boldsymbol{\zeta}_{\boldsymbol{t}}(j) = 0.$$

This result implies that for any $j = 1, \ldots, p$, we have $\lim_{t_j \downarrow 0} \partial f_\lambda(\boldsymbol{t})/\partial t_j = \lambda$, where $\lim_{t_j \downarrow 0}$ denotes the existence of the limit for any sequence of t_j that converges to 0 from the right. Since $\boldsymbol{\zeta}_{\boldsymbol{t}}$ is independent of λ, the above limit implies that there is a window $(0, a_j)$ such that the slope $\partial f_\lambda(\boldsymbol{t})/\partial t_j > 0$ for $t_j \in (0, a_j)$ and also the window size increases (i.e., a_j increases) as λ increases. As a result, for the function $g_\lambda(\boldsymbol{w})$, there exists a constant $a_j' > 0$ such that

$$\frac{\partial g_\lambda(\boldsymbol{w})}{\partial w_j} \begin{cases} < 0, & \text{for } -a_j' < w_j < 0 \\ > 0, & \text{for } 0 < w_j < a_j'. \end{cases}$$

In other words, for positive λ, there is a 'valley' on the surface of $g_\lambda(\boldsymbol{w})$ along the line $w_j = 0$ and the valley becomes wider as λ increases. In summary, the larger the values of λ the more w_j (or, equivalently t_j) have tendency to move towards 0 by the optimization algorithm and then a sparse model is selected (i.e, small number k of variables chosen). At the extreme value $\lambda_{\max} = \|y\|_2^2 / n$, all t_j are forced towards 0 and thus the null model will be selected.

6 Efficient Implementation of COMBSS

In this section, we focus on efficient implementation of COMBSS using the *conjugate gradient* method, the *Woodbury matrix identity*, and the *Banachiewicz Inversion Formula*.

6.1 Low- vs High-dimension

Recall the expression of $L_{\boldsymbol{t}}$ from (5):

$$L_{\boldsymbol{t}} = \frac{1}{n} \left[X_{\boldsymbol{t}}^{\mathsf{T}} X_{\boldsymbol{t}} + \delta \left(I - T_{\boldsymbol{t}}^2 \right) \right].$$

We have noticed earlier from Theorem 4 that for computing $\nabla g_\lambda(\boldsymbol{w})$, twice we evaluate matrix-vector products of the form $L_{\boldsymbol{t}}^{-1}\boldsymbol{u}$, which is the unique solution of the linear equation $L_{\boldsymbol{t}}\boldsymbol{z} = \boldsymbol{u}$. Solving linear equations efficiently is one of the important and well-studied problems in the field of linear algebra. Among many elegant approaches for solving linear equations, the conjugate gradient method is well-suited for our problem as $L_{\boldsymbol{t}}$ is symmetric positive-definite; see, for example, Golub and Van Loan (1996).

The running time of the conjugate gradient method for solving the linear equation $A\boldsymbol{z} = \boldsymbol{u}$ depends on the dimension of A. For our algorithm, since $L_{\boldsymbol{t}}$ is of dimension $p \times p$, the conjugate gradient method can return a good approximation of $L_{\boldsymbol{t}}^{-1}\boldsymbol{u}$ within $O(p^2)$ time by fixing the maximum number of iterations taken by the conjugate gradient method. This is true for both low-dimensional models (where $p < n$) and high-dimensional models (where $n < p$).

We now specifically focus on high-dimensional models and transform the problem of solving the p-dimensional linear equation $L_{\boldsymbol{t}}\boldsymbol{z} = \boldsymbol{u}$ to the problem of solving an n-dimensional linear equation problem. This approach is based on a well-known result in linear algebra called the Woodbury matrix identity. Since we are calling

the gradient descent method for solving a n-dimensional problem, instead of p-dimensional, we can achieve a much lower overall computational complexity for the high-dimensional models. The following result is a consequence of the Woodbury matrix identity, which is stated as Lemma 2 in Appendix A.

Theorem 6 *For $\boldsymbol{t} \in [0,1)^p$, let $S_{\boldsymbol{t}}$ be a p-dimensional diagonal matrix with the jth diagonal element being $n/\delta(1-t_j^2)$ and $\widetilde{L}_{\boldsymbol{t}} = I + X_{\boldsymbol{t}} S_{\boldsymbol{t}} X_{\boldsymbol{t}}^{\mathsf{T}}/n$. Then,*

$$L_{\boldsymbol{t}}^{-1}\boldsymbol{u} = (S_{\boldsymbol{t}}\boldsymbol{u}) - \frac{1}{n} S_{\boldsymbol{t}} X_{\boldsymbol{t}}^{\mathsf{T}} \widetilde{L}_{\boldsymbol{t}}^{-1} (X_{\boldsymbol{t}} S_{\boldsymbol{t}} \boldsymbol{u}).$$

The above expression suggests that instead of solving the p-dimensional problem $L_{\boldsymbol{t}}\boldsymbol{z} = \boldsymbol{u}$ directly, we can first solve the n-dimensional problem $\widetilde{L}_{\boldsymbol{t}}\boldsymbol{z} = (X_{\boldsymbol{t}} S_{\boldsymbol{t}} \boldsymbol{u})$ and substitute the result in the above expression to get the value of $L_{\boldsymbol{t}}^{-1}\boldsymbol{u}$.

6.2 A Dimension Reduction Approach

During the execution of the gradient descent algorithm, Step 1 of Algorithm 1, some of w_j (and hence the corresponding t_j) can reach zero. Particularly, for basic gradient descent and similar methods, once w_j reaches zero it remains zero until the algorithm terminates, because the update of $\boldsymbol{w}$ in the lth iteration of the basic gradient descent depends only on the gradient $g_\lambda(\boldsymbol{w}^{(l)})$, whose jth element

$$\frac{\partial g_\lambda(\boldsymbol{w}^{(l)})}{\partial w_j} = 0 \quad \text{if } w_j^{(l)} = 0. \tag{14}$$

Because (14) holds, we need to only focus on $\partial g_\lambda(\boldsymbol{w})/\partial w_j$ associated with $w_j \neq 0$ in order to reduce the cost of computing the gradient $\nabla g_\lambda(\boldsymbol{w})$. To simplify the notation, let $\mathscr{P} = \{1,\ldots,p\}$ and for any $\boldsymbol{t} \in [0,1)^p$, let $\mathscr{Z}_{\boldsymbol{t}}$ be the set of indices of the zero elements of $\boldsymbol{t}$, that is,

$$\mathscr{Z}_{\boldsymbol{t}} = \{j : t_j = 0, j = \mathscr{P}\}. \tag{15}$$

Similar to the notation used in Theorem 2, for a vector $\boldsymbol{u} \in \mathbb{R}^p$, we write $(\boldsymbol{u})_\star$ (respectively, $(\boldsymbol{u})_0$) to denote the vector of dimension $p - |\mathscr{Z}_{\boldsymbol{t}}|$ (respectively, $|\mathscr{Z}_{\boldsymbol{t}}|$) constructed from $\boldsymbol{u}$ by removing all its elements with the indices in $\mathscr{Z}_{\boldsymbol{t}}$ (respectively, in $\mathscr{P} \setminus \mathscr{Z}_{\boldsymbol{t}}$). Similarly, for a matrix A of dimension $p \times p$, we write $(A)_\star$ (respectively, $(A)_0$) to denote the new matrix constructed from A by removing its rows and columns with the indices in $\mathscr{Z}_{\boldsymbol{t}}$ (respectively, in $\mathscr{P} \setminus \mathscr{Z}_{\boldsymbol{t}}$). Then we have the following result.

Theorem 7 *Suppose $\boldsymbol{t} \in [0,1)^p$. Then,*

$$(L_{\boldsymbol{t}})_\star = \frac{1}{n}\left[(T_{\boldsymbol{t}})_\star \left(X^{\mathsf{T}}X\right)_\star (T_{\boldsymbol{t}})_\star + \delta\left(I - (T_{\boldsymbol{t}})_\star\right)\right].$$

Furthermore, we have

$$\begin{aligned} \left(L_{\boldsymbol{t}}^{-1}\right)_0 &= \frac{n}{\delta} I, \\ \left(L_{\boldsymbol{t}}^{-1}\right)_\star &= \left((L_{\boldsymbol{t}})_\star\right)^{-1}, \end{aligned} \tag{16}$$

$$\begin{aligned} \left(\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}\right)_0 &= \boldsymbol{0}, \\ \left(\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}\right)_\star &= \left((L_{\boldsymbol{t}})_\star\right)^{-1} \left((\boldsymbol{t})_\star \odot \left(\frac{X^{\mathsf{T}}\boldsymbol{y}}{n}\right)_\star\right), \end{aligned} \tag{17}$$

$$\begin{aligned} (\boldsymbol{c}_{\boldsymbol{t}})_0 &= \boldsymbol{0}, \\ (\boldsymbol{c}_{\boldsymbol{t}})_\star &= \left((L_{\boldsymbol{t}})_\star\right)^{-1} \left((\boldsymbol{t})_\star \odot (\boldsymbol{a}_{\boldsymbol{t}})_\star\right). \end{aligned} \tag{18}$$

In Theorem 7, (16) shows that for every $j \in \mathscr{Z}_{\boldsymbol{t}}$, all the off-diagonal elements of the jth row as well as the jth column of $L_{\boldsymbol{t}}^{-1}$ are zero while its jth diagonal element is n/δ, and all other elements of $L_{\boldsymbol{t}}^{-1}$ (which constitute the sub-matrix $\left(L_{\boldsymbol{t}}^{-1}\right)_\star$) depend only on $(L_{\boldsymbol{t}})_\star$, which can be computed using only the columns of the design matrix X with indices in $\mathscr{P} \setminus \mathscr{Z}_{\boldsymbol{t}}$. As a consequence, (17) and (18) imply that computing $\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}$ and $\boldsymbol{c}_{\boldsymbol{t}}$ is equal to solving $p_\star$-dimensional linear equations of the form $\left(L_{\boldsymbol{t}}^{-1}\right)_\star \boldsymbol{z} = \boldsymbol{v}$, where $p_\star = p - |\mathscr{Z}_{\boldsymbol{t}}|$. Since $p_\star \leq p$, solving such a $p_\star$-dimensional linear equation using the conjugate gradient can be faster than solving the original p-dimensional linear equation of the form $L_{\boldsymbol{t}}\boldsymbol{z} = \boldsymbol{u}$.

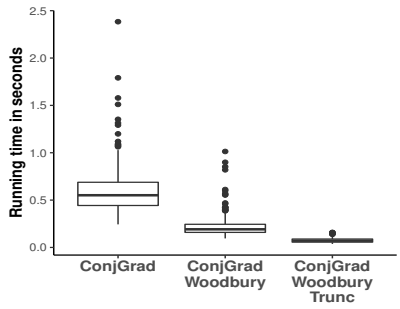
In summary, for a vector $\boldsymbol{t} \in [0,1)^p$ with some elements being 0, the values of $f_\lambda(\boldsymbol{t})$ and $\nabla f_\lambda(\boldsymbol{t})$ do not depend on the columns j of X where $t_j = 0$. Therefore, we can reduce the computational complexity by removing all the columns j of the design matrix X where $t_j = 0$.

6.3 Making Our Algorithm Fast

In Section 6.2, we noted that when some elements t_j of $\boldsymbol{t}$ are zero, it is faster to compute the objective functions $f_\lambda(\boldsymbol{t})$ and $g_\lambda(\boldsymbol{t})$ and their gradients $\nabla f_\lambda(\boldsymbol{t})$ and $\nabla g_\lambda(\boldsymbol{t})$ by ignoring the columns j of the design matrix X. In Section 5.2, using Proposition 5, we further noted that for any $\lambda > 0$ there is a 'valley' on the surface of $g_\lambda(\boldsymbol{w})$ along $w_j = 0$ for all $j = 1, \ldots, p$, and thus for any j, when w_j (or, equivalently, t_j) is sufficiently small during the execution of the gradient descent method, it will eventually become zero. Using these observations, in the implementation of our method, to reduce the computational cost of estimating the gradients, it is wise to map w_j (and t_j) to 0 when w_j is almost zero. We incorporate this truncation idea into our algorithm as follows.

We first fix a small constant η, say at 0.001. As we run the gradient descent algorithm, when t_j becomes smaller than η for some $j \in \mathscr{P}$, we take t_j and w_j to be zero and we stop updating them; that is, t_j and w_j will continue to be zero until the gradient descent algorithm terminates. In each iteration of the gradient descent algorithm, the design matrix is updated by removing all the columns corresponding to zero t_j's. If the algorithm starts at $\boldsymbol{w}$ with all non-zero elements, the effective dimension $p_\star$, which denotes the number of columns in the updated design matrix, monotonically decreases starting from p. In an iteration, if $p_\star > n$, we can use Theorem 6 to reduce the complexity of computing the gradients. However, when $p_\star$ falls below n, we directly use conjugate gradient for computing the gradients without invoking Theorem 6.

Using a dataset, Fig. 4 illustrates the substantial improvement in the speed of our algorithm when the above mentioned improvement ideas are incorporated in its implementation.



(a) $p = 1000$

(b) $p = 5000$

Fig. 4: Running time of our algorithm at $\lambda = 0.1$ for an example dataset using the adam optimizer, a popular gradient based method. These boxplots are based on 300 replications. Here we compare running times for COMBSS with SubsetMapV1 using only conjugate gradient (ConjGrad), conjugate gradient with Woodbury matrix identity (ConjGrad-Woodbury), and conjugate gradient with both Woodbury matrix identity and truncation improvement (ConjGrad-Woodbury-Trunc). For the truncation, $\eta = 0.001$. The dataset is generated using the Beta-type 1 model presented in Section 7 of (Moka et al, 2022) with $k_0 = 5$, $n = 100$, $\rho = 0.8$, and the signal-to-noise ratio of 5.

7 Simulation Experiments

Our method is available through Python and R codes provided at https://github.com/saratmoka/COMBSS-Python-VIGNETTE and https://github.com/benoit-liquet/COMBSS-R-VIGNETTE, respectively. The code includes examples where p is as large as of order 10,000. This code further allows to replicate our empirical research in this article and the simulation studies presented in Moka et al (2022) where efficacy of our method is compared to three popular existing methods, namely, forward selection (FS), Lasso, and mixed integer optimization (MIO). In Moka et al (2022), we focused on demonstrating (using SubsetMapV1) the efficacy in predicting the true model of the data.

Here, our focus is on demonstrating the efficacy of our method in retrieving best subsets of given sizes, meaning our ability to solve (1) using SubsetMapV2. We also compare our approach to forward selection, Lasso, mixed integer optimization and L0Learn (Hazimeh and Mazumder, 2020b).

7.1 Simulation design

The data is generated from the linear model:

$$\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad \text{where } \epsilon \sim N_n\left(0, \sigma^2 \mathbb{I}_n\right), \quad (19)$$

where each row of the predictor matrix X is generated from a multivariate normal distribution with zero mean and covariance matrix Σ with diagonal elements $\Sigma_{j,j} = 1$ and off-diagonal elements $\Sigma_{i,j} = \rho^{|i-j|}$, $i \neq j$, for some correlation parameter $\rho \in (-1, 1)$. In order to investigate a challenging situation, we use $\rho = 0.8$ to mimic strong correlation between predictors. For each simulation, we fix the signal-to-noise ratio (SNR) and compute the variance σ^2 of the noise $\boldsymbol{\epsilon}$ using

$$\sigma^2 = \frac{\boldsymbol{\beta}^\intercal \Sigma \boldsymbol{\beta}}{\mathsf{SNR}}.$$

We consider the following two simulation settings:

- **Beta-type 2:** The first $k_0 = 10$ components of $\boldsymbol{\beta}$ are equal to 1 and all other components of $\boldsymbol{\beta}$ are equal to 0.
- **Beta-type 3:** The first $k_0 = 10$ components of $\boldsymbol{\beta}$ are equal to $\beta_i = 0.5^{i-1}$, for $i = 1, \ldots, k_0$ and all other components of $\boldsymbol{\beta}$ are equal to 0.

Both Beta-type 2 and Beta-type 3 assumes strong correlation between the active predictors. Beta-type 3 differs from Beta-type 2 by presenting a signal decaying exponentially to 0. Beta-type 2 has been used in Moka et al (2022) and so to avoid confusion we call the decay signal as Beta-type 3.

For both these types, we investigate the performance of our method in low- and high-dimensional settings. For the low-dimensional setting, we take $n = 100$ and $p = 20$ for SNR $\in \{0.5, 1, 2, \ldots, 8\}$, while for the high-dimensional setting, $n = 100$ and $p = 1000$ for SNR $\in \{2, 3, \ldots, 8\}$.

In the low-dimensional setting, FS and MIO were tuned over $k = 0, \ldots, 20$. In this simulation we ran MIO through the R package `bestsubest` offered in Hastie et al (2018) while we ran L0Learn through the R package `L0Learn` offered in Hazimeh et al (2023). For the high dimensional setting, we do not include MIO due to time computational constraints posed by MIO. In particular, the MIO based on the Gurobi optimizer is quite time consuming for high dimensional cases (see Hastie et al (2020)).

In low- and high-dimensional settings, the Lasso was tuned for 50 values of λ ranging from $\lambda_{\max} = \left\|X^T \boldsymbol{y}\right\|_\infty$ to a small fraction of $\lambda_{\max}$ on a log scale, as per the default in `bestsubest` package. In both the low- and high-dimensional settings, COMBSS with SubsetMapV2 was called four times starting at four different initial points $\boldsymbol{t}^{(0)}$: $(0.5, \ldots, 0.5)^\intercal$, $(0.99, \ldots 0.99)^\intercal$, $(0.75, \ldots, 0.75)^\intercal$, and $(0.3, \ldots, 0.3)^\intercal$. For each call we used 24 values of λ on a grid as follows. Starting from $\lambda_{\max} = \|\boldsymbol{y}\|_2^2 / n$, half of λ values were generated by $\{\lambda_l = \lambda_{\max}/2^{(l)}, \; l = 1, \ldots, 12\}$. From this sequence, the remaining λ values were created by $\{(\lambda_{l+1} + \lambda_l)/2 : l = 1, \ldots, 12\}$.

7.2 Low-dimensional case

In low dimensional case, we use the exhaustive method to find the exact solution of the best subset for any subset size ranging from 1 to p. Then, we assess our method in retrieving the exact best subset for each subset size. Figure 5, shows the frequency (over 50 replications) of retrieving the exact best subset (provided by exhaustive search) for any subset size from $k = 1, \ldots, p$. For each SNR level, MIO as expected retrieves perfectly the *true* best subset of any model size. Then COMBSS gives the best results to retrieve the best subset compared to FS, Lasso and L0Learn. Similar behaviours are reported for Beta-type 3 in Figure 6.

7.3 High-dimensional case

Note that the exact best subset is unknown for the high dimensional case since it is computationally impractical to conduct an exhaustive search over all the subsets of sizes 1 to $p = 1000$. Hence, to assess the performance of our method in retrieving a *competing* best subset, we compare the best subset obtained from COMBSS for two different subset sizes: 5 and 10. For this comparison, we use

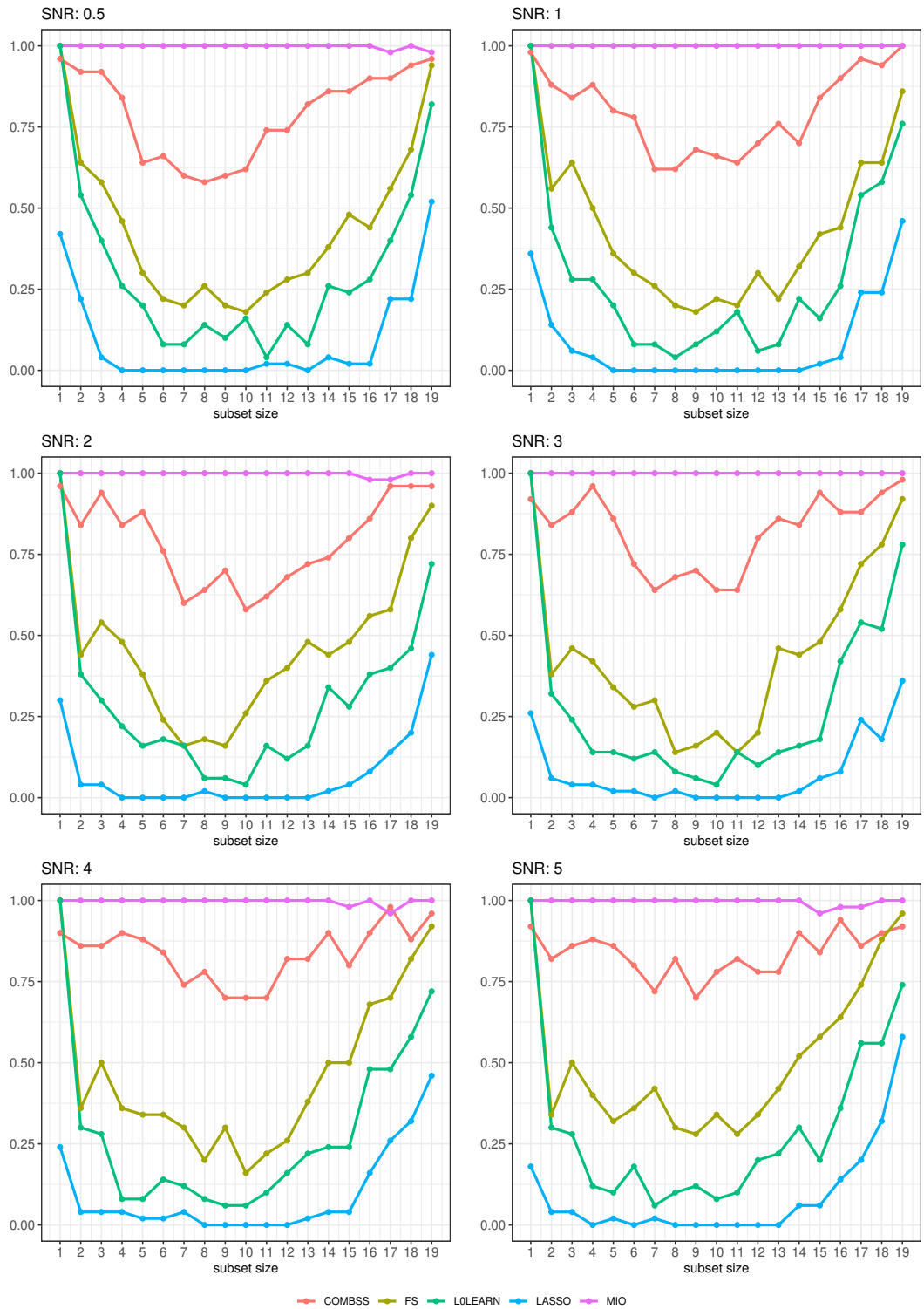


Fig. 5: Frequency (over 50 replications) of retrieving the exact best subset for any subset size from $k = 1, \ldots, p$ for beta-type 2 case.

Fig. 6: Frequency (over 50 replications) of retrieving the exact best subset for any subset size from $k = 1, \ldots, p$ for beta-type 3 case.

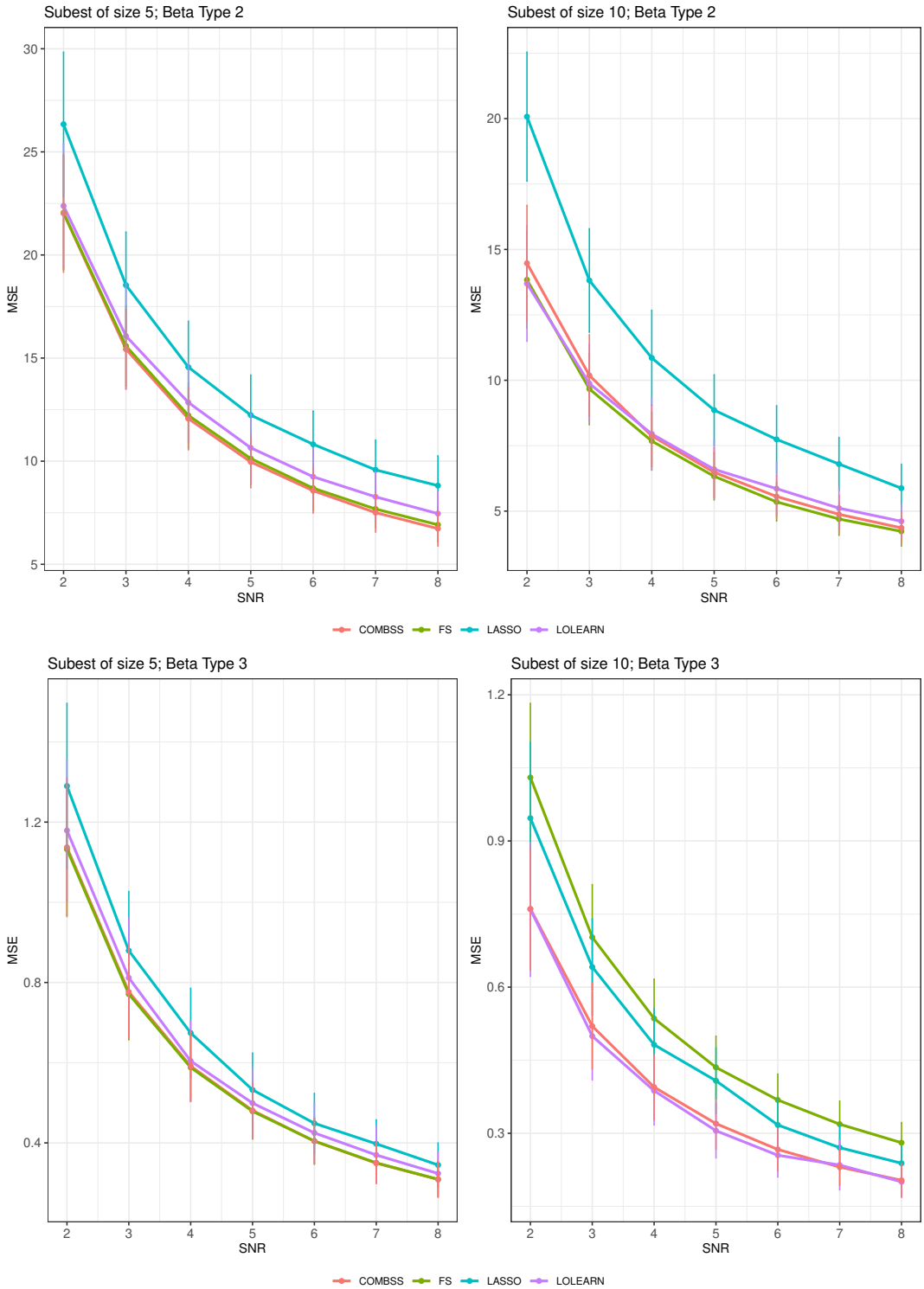


Fig. 7: Ability of COMBSS for providing a *competing* best subset for subset size 5 and 10 comparing to FS, Lasso and L0Learn. Top plots are for beta-type 2 and bottom plots correspond to beta-type 3.

the mean square of error (MSE) of the dataset to evaluate which method is providing a better subset for size 5 and 10. Figure 7 presents these results over 50 replications for SNR values from 2 to 8. Overall, COMBSS is consistently same or better than other methods for providing a *competing* best subset. On the other hand none of the alternative methods is consistent across all the cases.

8 Conclusion and Discussion

In this paper, we have introduced COMBSS, a novel continuous optimization method towards best subset selection in linear regression. The key goal of COMBSS is to extend the highly difficult discrete constrained best subset selection problem to an unconstrained continuous optimization problem. In particular, COMBSS involves extending the objective function of the best subset selection, which is defined at the corners of the hypercube $[0,1]^p$, to a differentiable function defined on the whole hypercube. For this extended function, starting from an interior point, a gradient descent method is executed to find a corner of the hypercube where the objective function is minimum.

In our algorithm, the primary operations involved are the matrix-vector product, the vector-vector element-wise product, and the scalar-vector product. Additionally, we note that the primary operation for executing a conjugate gradient method for solving a linear equation $A\boldsymbol{z} = \boldsymbol{u}$ is the matrix-vector product $A\boldsymbol{u}$. All these operations are known to execute faster on graphics processing unit based computers using parallel programming, which could substantially increase the speed of our method further.

In the preprint Moka et al (2022), we have conducted several simulation experiments in both low-dimensional and high-dimensional setups to illustrate the good performance of COMBSS with SubsetMapV1 for predicting the true model of the data in comparison to the existing popular methods: Forward Stepwise, Lasso, and Mixed Integer Optimization. In this paper, our simulation experiments highlight the ability of COMBSS with SubsetMapV2 for retrieving "exact" best subset for any subset size in comparison to the existing methods. Both of these empirical results highlight the potential of COMBSS for feature extractions.

Our simulation experiments show that the sequence of vectors $\boldsymbol{t}^{(1)}, \boldsymbol{t}^{(2)}, \ldots$ obtained during the execution of gradient descent always converges. A future work focusing on the theoretical study for establishing such convergence would be useful.

A future direction for finding the best model of a given fixed size k is to explore different options for the penalty term of the objective function $f_\lambda(\boldsymbol{t})$. Ideally, if we select a sufficiently large penalty for $\sum_{j=1}^p t_j > k$ and 0 otherwise, we can drive the optimization algorithm towards a model of size k that lies along the hyperplane given by $\sum_{j=1}^p t_j = k$. Because such a discrete penalty is not differentiable, we could use smooth alternatives. For instance, the penalty could be taken to be $\lambda(k - \sum_{j=1}^p t_j)^2$ when $\sum_{j=1}^p t_j > k$ and 0 otherwise, for a tuning parameter $\lambda > 0$.

We expect, similarly to the significant body of work that focuses on the Lasso and on MIO, respectively, that there are many avenues that can be explored and investigated for building on the presented COMBSS framework. Particularly, to tackle best subset selection when problems are ultra high dimensional. In this paper, we have opened a novel framework for feature selection and this framework can be extended to other models beyond the linear regression model. For instance, recently Mathur et al (2023) extended the COMBSS framework for solving column subset selection and Nyström approximation problems. Furthermore, our ongoing research focuses on the extensions of COMBSS to non-linear regression problems including logistic regression.

Acknowledgements. Samuel Muller was supported by the Australian Research Council Discovery Project Grant #210100521.

Appendix A Proofs

Proof of Theorem 1 Since both $X_{\boldsymbol{t}}^\top X_{\boldsymbol{t}}$ and $T_{\boldsymbol{t}}$ are symmetric, the symmetry of $L_{\boldsymbol{t}}$ is obvious. We now show that $L_{\boldsymbol{t}}$ is positive-definite for $\boldsymbol{t} \in [0,1)^p$ by establishing

$$\boldsymbol{u}^\top L_{\boldsymbol{t}} \boldsymbol{u} > 0, \quad \text{for all } \boldsymbol{u} \in \mathbb{R}^p \setminus \{\boldsymbol{0}\}. \tag{A1}$$

The matrix $X_{\boldsymbol{t}}^\top X_{\boldsymbol{t}}$ is a positive semi-definite, because

$$\boldsymbol{u}^\top X_{\boldsymbol{t}}^\top X_{\boldsymbol{t}} \boldsymbol{u} = \|X_{\boldsymbol{t}} \boldsymbol{u}\|_2^2 \geq 0.$$

In addition, for all $\boldsymbol{t} \in [0,1)^p$, the matrix $\delta\left(I - T_{\boldsymbol{t}}^2\right)$ is also a positive-definite because $\delta > 0$, and

$$\boldsymbol{u}^{\top}\left(I - T_{\boldsymbol{t}}^2\right)\boldsymbol{u} = \|\boldsymbol{u}\|_2^2 - \|T_{\boldsymbol{t}}\boldsymbol{u}\|_2^2 = \sum_{j=1}^{p} u_j^2(1-t_j^2), \tag{A2}$$

which is strictly positive if $\boldsymbol{t} \in [0,1)^p$ and $\boldsymbol{u} \in \mathbb{R}^p \backslash \{\boldsymbol{0}\}$. Since positive-definite matrices are invertible, we have $L_{\boldsymbol{t}}^{\dagger} = L_{\boldsymbol{t}}^{-1}$, and thus, $\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}} = L_{\boldsymbol{t}}^{-1} X_{\boldsymbol{t}}^{\top}\boldsymbol{y}/n$. □

Theorem 8 is a collection of results from the literature that we need in our proofs. Results (*i*) and (*ii*) of Theorem 8 are well-known in the literature as Banachiewicz inversion lemma (see, e.g., Tian and Takane (2005)), and (*iii*) is its generalization to Moore–Penrose inverse (See Corollary 3.5 (c) in Castro-González et al (2015)).

Theorem 8 *Let M be a square block matrix of the form*

$$M = \begin{bmatrix} A & C \\ B & D \end{bmatrix}$$

with A being a square matrix. Let the Schur complement $S = D - BA^{\dagger}C$. Suppose that D is non-singular. Then following holds.

(i) If A is non-singular, then M is non-singular if and only if S is non-singular.

(ii) If both A and S are non-singular, then

$$M^{-1} = \begin{bmatrix} I & -A^{-1}CS^{-1} \\ 0 & S^{-1} \end{bmatrix}\begin{bmatrix} A^{-1} & 0 \\ -BA^{-1} & I \end{bmatrix} = \begin{bmatrix} I & -A^{-1}C \\ 0 & I \end{bmatrix}\begin{bmatrix} I & 0 \\ -S^{-1}B & S^{-1} \end{bmatrix}\begin{bmatrix} A^{-1} & 0 \\ 0 & I \end{bmatrix}. \tag{A3}$$

(iii) If A is singular, S is non-singular, $BA^{\dagger}A = B$ and $AA^{\dagger}C = C$, then

$$M^{\dagger} = \begin{bmatrix} I & -A^{\dagger}CS^{-1} \\ 0 & S^{-1} \end{bmatrix}\begin{bmatrix} A^{\dagger} & 0 \\ -BA^{\dagger} & I \end{bmatrix} = \begin{bmatrix} I & -A^{\dagger}C \\ 0 & I \end{bmatrix}\begin{bmatrix} I & 0 \\ -S^{-1}B & S^{-1} \end{bmatrix}\begin{bmatrix} A^{\dagger} & 0 \\ 0 & I \end{bmatrix}. \tag{A4}$$

Proof of Theorem 2 The inverse of a matrix after a permutation of rows (respectively, columns) is identical to the matrix obtained by applying the same permutation on columns (respectively, rows) on the inverse of the matrix. Therefore, without loss of generality, we assume that all the zero-elements of $\boldsymbol{s} \in \{0,1\}^p$ appear at the end, in the form:

$$\boldsymbol{s} = (s_1, \ldots, s_m, 0, \ldots, 0),$$

where m indicates the number of non-zeros in $\boldsymbol{s}$. Recall that $X_{[\boldsymbol{s}]}$ is the matrix of size $n \times |\boldsymbol{s}|$ created by keeping only columns j of X for which $s_j = 1$. Thus, $L_{\boldsymbol{s}}$ is given by,

$$L_{\boldsymbol{s}} = \frac{1}{n}\left[\begin{pmatrix} X_{[\boldsymbol{s}]}^{\top}X_{[\boldsymbol{s}]} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix} + \delta\begin{pmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & I \end{pmatrix}\right] = \frac{1}{n}\begin{pmatrix} X_{[\boldsymbol{s}]}^{\top}X_{[\boldsymbol{s}]} & \boldsymbol{0} \\ \boldsymbol{0} & \delta I \end{pmatrix}. \tag{A5}$$

From Theorem 8 (i), it is evident that $L_{\boldsymbol{s}}$ is invertible if and only if $X_{[\boldsymbol{s}]}^{\top}X_{[\boldsymbol{s}]}$ is invertible.

First assume that $X_{[\boldsymbol{s}]}^{\top}X_{[\boldsymbol{s}]}$ is invertible. Then, from Theorem 8 (ii),

$$L_{\boldsymbol{s}}^{-1} = n\begin{pmatrix} I & \boldsymbol{0} \\ \boldsymbol{0} & \frac{1}{\delta}I \end{pmatrix}\begin{pmatrix} \left(X_{[\boldsymbol{s}]}^{\top}X_{[\boldsymbol{s}]}\right)^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & I \end{pmatrix} = n\begin{pmatrix} \left(X_{[\boldsymbol{s}]}^{\top}X_{[\boldsymbol{s}]}\right)^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \frac{1}{\delta}I \end{pmatrix}. \tag{A6}$$

Now recall the notations $(\widetilde{\boldsymbol{\beta}}_{\boldsymbol{s}})_+$ and $(\widetilde{\boldsymbol{\beta}}_{\boldsymbol{s}})_0$ introduced before stating Theorem 2. Then, we use (A6) to obtain

$$\begin{pmatrix} (\widetilde{\boldsymbol{\beta}}_{\boldsymbol{s}})_+ \\ (\widetilde{\boldsymbol{\beta}}_{\boldsymbol{s}})_0 \end{pmatrix} = L_{\boldsymbol{s}}^{-1}T_{\boldsymbol{s}}\frac{X^{\top}\boldsymbol{y}}{n} = \begin{pmatrix} \left(X_{[\boldsymbol{s}]}^{\top}X_{[\boldsymbol{s}]}\right)^{-1}X_{[\boldsymbol{s}]}^{\top}\boldsymbol{y} \\ \boldsymbol{0} \end{pmatrix}.$$

This further guarantees that $X_{[\boldsymbol{s}]}\widehat{\boldsymbol{\beta}}_{[\boldsymbol{s}]} = X_{\boldsymbol{s}}\widetilde{\boldsymbol{\beta}}_{\boldsymbol{s}}$.

When $X_{[\boldsymbol{s}]}^{\top}X_{[\boldsymbol{s}]}$ is singular, by replacing the inverse with its pseudo-inverse in the above discussion, and using Theorem 8 (iii) instead of Theorem 8 (ii), we can establish the same conclusions. This is because, the corresponding Schur complement for $L_{\boldsymbol{s}}$ is $S = n\,I/\delta$, which is symmetric and positive definite.

□

Proof of Theorem 3 Consider a sequence $\boldsymbol{t}_1, \boldsymbol{t}_2, \cdots \in [0,1)^p$ that converges a point $\boldsymbol{t} \in [0,1]^p$. We know that the converges easily holds when $\boldsymbol{t} \in [0,1)^p$ from the continuity of matrix inversion which states that for any sequence of invertible matrices $Z_1, Z_2, \ldots$ that converging to an invertible matrix Z, the sequence of their inverses $Z_1^{-1}, Z_2^{-1}, \ldots$ converges to Z^{-1}.

Now using Theorem 8, we prove the convergence when some or all of the elements of $\boldsymbol{t}$ are equal to 1. Suppose mt has exactly m elements equal to 1. Using the arguments from the proof of 2, without of loss of

generality assume that all 1s in $\boldsymbol{t}$ appear together in the first m positions, that is,

$$\boldsymbol{t} = (\underbrace{1, \ldots, 1}_{m \text{ times}}, \underbrace{t_{m+1}, \ldots, t_p}_{p-m \text{ times}}).$$

In that case, by writing

$$T_{\ell,1} = \text{Diag}\,(t_{\ell,1}, \ldots, t_{\ell,m}), \quad \text{and}$$
$$T_{\ell,2} = \text{Diag}\,(t_{\ell,m+1}, \ldots, t_{\ell,p}),$$

we observe that as $\ell \to \infty$

$$T_{\ell,1} \longrightarrow I, \quad \text{and} \quad T_{\ell,2} \longrightarrow T_2 = \text{Diag}\,(t_{m+1}, \ldots, t_p).$$

Further, take

$$F_\ell = \begin{bmatrix} T_{\ell,1} & 0 \\ 0 & I \end{bmatrix},$$

and

$$X = [X_1, X_2],$$

with X_1 denoting the first m columns of X. Similarly, we can write

$$X_{\boldsymbol{t}_\ell} = [X_{\boldsymbol{t}_\ell,1}, X_{\boldsymbol{t}_\ell,2}].$$

We now observe that

$$\begin{aligned} X_{\boldsymbol{t}_\ell}^\top X_{\boldsymbol{t}_\ell} &= \begin{bmatrix} X_{\boldsymbol{t}_\ell,1}^\top X_{\boldsymbol{t}_\ell,1} & X_{\boldsymbol{t}_\ell,1}^\top X_{\boldsymbol{t}_\ell,2} \\ X_{\boldsymbol{t}_\ell,2}^\top X_{\boldsymbol{t}_\ell,1} & X_{\boldsymbol{t}_\ell,2}^\top X_{\boldsymbol{t}_\ell,2} \end{bmatrix} \\ &= \begin{bmatrix} T_{\ell,1} X_1^\top X_1 T_{\ell,1} & T_{\ell,1} X_1^\top X_{\boldsymbol{t}_\ell,2} \\ X_{\boldsymbol{t}_\ell,2}^\top X_1 T_{\ell,1} & X_{\boldsymbol{t}_\ell,2}^\top X_{\boldsymbol{t}_\ell,2} \end{bmatrix} \\ &= F_\ell \begin{bmatrix} X_1^\top X_1 & X_1^\top X_{\boldsymbol{t}_\ell,2} \\ X_{\boldsymbol{t}_\ell,2}^\top X_1 & X_{\boldsymbol{t}_\ell,2}^\top X_{\boldsymbol{t}_\ell,2} \end{bmatrix} F_\ell. \end{aligned}$$

As a result,

$$\begin{aligned} L_{\boldsymbol{t}_\ell} &= X_{\boldsymbol{t}_\ell}^\top X_{\boldsymbol{t}_\ell} + \delta(I - T_{\boldsymbol{t}_\ell}^2) \\ &= F_\ell \left(\begin{bmatrix} X_1^\top X_1 & X_1^\top X_{\boldsymbol{t}_\ell,2} \\ X_{\boldsymbol{t}_\ell,2}^\top X_1 & X_{\boldsymbol{t}_\ell,2}^\top X_{\boldsymbol{t}_\ell,2} \end{bmatrix} \right. \\ &\qquad \left. + \delta \begin{bmatrix} T_{\boldsymbol{t}_\ell,1}^{-2} - I & 0 \\ 0 & I - T_{\boldsymbol{t}_\ell,2}^2 \end{bmatrix} \right) F_\ell. \end{aligned}$$

Now define,

$$\begin{aligned} A_\ell &= X_1^\top X_1 + \delta(T_{\boldsymbol{t}_\ell,1}^{-2} - I), \\ B_\ell &= X_{\boldsymbol{t}_\ell,2}^\top X_1, \\ C_\ell &= X_1^\top X_{\boldsymbol{t}_\ell,2}, \\ D_\ell &= X_{\boldsymbol{t}_\ell,2}^\top X_{\boldsymbol{t}_\ell,2} + \delta(I - T_{\boldsymbol{t}_\ell,2}^2), \end{aligned}$$

and

$$M_\ell = \begin{bmatrix} A_\ell & C_\ell \\ B_\ell & D_\ell \end{bmatrix}.$$

Since $\boldsymbol{t}_\ell \in [0,1)^p$, $L_{\boldsymbol{t}_\ell}$ is non-singular (see Theorem 1), and hence we have

$$L_{\boldsymbol{t}_\ell}^{-1} = F_\ell^{-1} M_\ell^{-1} F_\ell^{-1}.$$

Note that the corresponding Schur complement $S_\ell = D_\ell - B_\ell A_\ell^{-1} C_\ell$ is non-singular from Theorem 8 (i). Furthermore, since

$$X_{\boldsymbol{t}_\ell} = \begin{bmatrix} X_1 & X_{\boldsymbol{t}_\ell,2} \end{bmatrix} F_\ell,$$

$$L_{\boldsymbol{t}_\ell}^{-1} X_{\boldsymbol{t}_\ell}^\top = F_\ell^{-1} M_\ell^{-1} \begin{bmatrix} X_1^\top \\ X_{\boldsymbol{t}_\ell,2}^\top \end{bmatrix},$$

and hence,

$$\begin{aligned} \lim_{\ell \to \infty} L_{\boldsymbol{t}_\ell}^{-1} X_{\boldsymbol{t}_\ell}^\top &= \lim_{\ell \to \infty} F_\ell \lim_{\ell \to \infty} \left(M_\ell^{-1} \begin{bmatrix} X_1^\top \\ X_{\boldsymbol{t}_\ell,2}^\top \end{bmatrix} \right) \\ &= \lim_{\ell \to \infty} \left(M_\ell^{-1} \begin{bmatrix} X_1^\top \\ X_{\boldsymbol{t}_\ell,2}^\top \end{bmatrix} \right). \end{aligned}$$

Using (A3),

$$M_\ell^{-1} = \begin{bmatrix} I & -A_\ell^{-1} C_\ell \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ -S_\ell^{-1} B_\ell & S_\ell^{-1} \end{bmatrix} \begin{bmatrix} A_\ell^{-1} & 0 \\ 0 & I \end{bmatrix},$$

and hence,

$$M_\ell^{-1} \begin{bmatrix} X_1^\top \\ X_{\boldsymbol{t}_\ell,2}^\top \end{bmatrix}$$

is equal to

$$\begin{bmatrix} I & -A_\ell^{-1} X_1^\top X_{\boldsymbol{t}_\ell,2} \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ -S_\ell^{-1} B_\ell & S_\ell^{-1} \end{bmatrix} \begin{bmatrix} A_\ell^{-1} X_1^\top & 0 \\ 0 & X_{\boldsymbol{t}_\ell,2}^\top \end{bmatrix}. \tag{A7}$$

Now by defining

$$\begin{aligned} A &= X_1^\top X_1, \\ B &= X_{\boldsymbol{t},2}^\top X_1, \\ C &= X_1^\top X_{\boldsymbol{t},2}, \quad \text{and} \\ D &= X_{\boldsymbol{t},2}^\top X_{\boldsymbol{t},2} + \delta(I - T_{\boldsymbol{t},2}^2), \end{aligned}$$

we have

$$L_{\boldsymbol{t}} = \begin{bmatrix} A & C \\ B & D \end{bmatrix}.$$

Since $T_{\boldsymbol{t},2} < I$, we can see that D is symmetric positive definite and hence non-singular (this can be established just like the proof of Theorem 1). Furthermore, the corresponding Schur complement $S = D - BA^\dagger C$ is symmetric positive definite, and hence non-singular. The symmetry of S is easy to see from the definition because A and D are symmetric and $B = C^\top$. To see that S is positive definite, for any $\boldsymbol{x} \in \mathbb{R}^{p-m} \setminus \{\boldsymbol{0}\}$, let $z = X_{\boldsymbol{t},2} x$ and thus

$$\begin{aligned} x^\top S x &= z^\top z + \delta x^\top (I - T_{\boldsymbol{t},2}^2) x - z^\top X_1 (X_1^\top X_1)^\dagger X_1^\top z \\ &> z^\top z - z^\top X_1 (X_1^\top X_1)^\dagger X_1^\top z \end{aligned}$$

$$= z^\top \left(I - X_1(X_1^\top X_1)^\dagger X_1^\top\right) z.$$

Since $\left(I - X_1(X_1^\top X_1)^\dagger X_1^\top\right)$ is a projection matrix and hence positive definite, S is also positive definite.

In addition, using the singular value decomposition (SVD) $X_1 = U_1\Delta_1 V_1^\top$, we have

$$\begin{aligned} BA^\dagger A &= X_{\boldsymbol{t},2}^\top X_1(X_1^\top X_1)^\dagger(X_1^\top X_1) \\ &= X_{\boldsymbol{t},2}^\top U_1\Delta_1(\Delta_1^\top\Delta_1)^\dagger(\Delta_1^\top\Delta_1)V_1 \\ &= X_{\boldsymbol{t},2}^\top U_1\Delta_1 V_1 \\ &= X_{\boldsymbol{t},2}^\top X_1 = B. \end{aligned}$$

Similarly, we can show that $AA^\dagger C = C$. Thus, using (A4), $L_{\boldsymbol{t}}^\dagger X_{\boldsymbol{t}}^\top$ is equal to

$$\begin{bmatrix} I & -A^\dagger X_1^\top X_{\boldsymbol{t},2} \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ -S^{-1}B & S^{-1} \end{bmatrix} \begin{bmatrix} A^\dagger X_1^\top & 0 \\ 0 & X_{\boldsymbol{t},2} \end{bmatrix}. \tag{A8}$$

Since $\lim_{\ell\to\infty} X_{\boldsymbol{t}_\ell,2} = X_{\boldsymbol{t},2}$ and $\lim_{\ell\to\infty} B_\ell = B$, from (A7) and (A8), to show that

$$\lim_{\ell\to\infty} L_{\boldsymbol{t}_\ell}^{-1} X_{\boldsymbol{t}_\ell}^\top = L_{\boldsymbol{t}}^\dagger X_{\boldsymbol{t}}^\top,$$

it is enough to show that

$$\lim_{\ell\to\infty} S_\ell^{-1} = S^{-1}, \tag{A9}$$

$$\lim_{\ell\to\infty} A_\ell^{-1} X_1^\top = A^\dagger X_1^\top. \tag{A10}$$

Since S and each of S_ℓ are non-singular, (A9) holds from the continuity of matrix inversion. Now observe that

$$\begin{aligned} A^\dagger X_1^\top &= \left(X_1^\top X_1\right)^\dagger X_1^\top \\ &= V_1\left(\Delta_1^\top\Delta_1\right)^\dagger \Delta_1^\top U_1^\top \\ &= V_1\Delta_1^\dagger U_1^\top \\ &= X_1^\dagger, \end{aligned}$$

To establish (A10), we need to show that $\overline{X} = \lim_{\ell\to\infty} A_\ell^{-1}X_1^\top$ is equal to $X_1^\dagger$. Towards this, define

$$\eta_\ell = \max_{i=1,\ldots,m}(1/t_{\ell,i}^2 - 1),$$

$$\epsilon_\ell = \min_{i=1,\ldots,m}(1/t_{\ell,i}^2 - 1).$$

Then, we observe that both η_ℓ and ϵ_ℓ are strictly positive and going to zero as $\ell\to\infty$. Thus,

$$X_1^\top X_1 + \delta\epsilon_\ell I \le A_\ell \le X_1^\top X_1 + \delta\eta_\ell I,$$

where for any two symmetric positive semi-definite matrices Z and Z', we write $Z \ge Z'$ if $Z - Z'$ is also positive semi-definite. Let

$$\underline{A}_\ell = X_1^\top X_1 + \delta\epsilon_\ell I \quad \text{and} \quad \overline{A}_\ell = X_1^\top X_1 + \delta\eta_\ell I.$$

Thus, $\underline{A}_\ell^{-1} \ge A_\ell^{-1} \ge \overline{A}_\ell^{-1}$, or, alternatively,

$$A_\ell^{-1} - \overline{A}_\ell^{-1} \le \underline{A}_\ell^{-1} - \overline{A}_\ell^{-1}.$$

Now for any matrix norm, denoting as $\|\cdot\|$, using the triangular inequality,

$$\begin{aligned} &\|A^{-1}X_1^\top - X_1^\dagger\| \\ &\quad = \|(A_\ell^{-1} - \overline{A}_\ell^{-1})X_1^\top + (\overline{A}_\ell^{-1}X_1^\top - X_1^\dagger)\| \\ &\quad \le \|(A_\ell^{-1} - \overline{A}_\ell^{-1})X_1^\top\| + \|(\overline{A}_\ell^{-1}X_1^\top - X_1^\dagger)\| \\ &\quad \le \|(\underline{A}_\ell^{-1} - \overline{A}_\ell^{-1})X_1^\top\| + \|(\overline{A}_\ell^{-1}X_1^\top - X_1^\dagger)\|. \end{aligned} \tag{A11}$$

Using the SVD of $X_1 = U_1\Delta_1 V_1^\top$, we get the SVD of $(\underline{A}_\ell^{-1} - \overline{A}_\ell^{-1})X_1^\top$ as

$$V_1\left((\Delta_1^\top\Delta_1 + \delta\epsilon_\ell I)^{-1}\Delta_1^\top - (\Delta_1^\top\Delta_1 + \delta\eta_\ell I)^{-1}\Delta_1^\top\right)U_1^\top.$$

That is, suppose σ_i is the ith singular value of X_1, then the ith singular value of $(\underline{A}_\ell^{-1} - \overline{A}_\ell^{-1})X_1^\top$ is 0 if $\sigma_i = 0$, otherwise, it is

$$\frac{\sigma_i}{\sigma_i^2 + \delta\epsilon_\ell} - \frac{\sigma_i}{\sigma_i^2 + \delta\eta_\ell} = \frac{\sigma_i\delta(\eta_\ell - \epsilon_\ell)}{(\sigma_i^2 + \delta\epsilon_\ell)(\sigma_i^2 + \delta\eta_\ell)},$$

which goes to zero and thus the first term in (A11) goes to zero. The second term in (A11) also converges to zero because of the limit definition of pseudo-inverse that states that for any matrix Z

$$Z^\dagger = \lim_{\epsilon\nearrow 0}\left(Z^\top Z + \epsilon I\right)^{-1} Z^\top.$$

This completes the proof. □

For proving Theorem 4, we use Lemma 1, which obtains the partial derivatives of $\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}$ with respect to the elements of $\boldsymbol{t}$.

Lemma 1 For any $\boldsymbol{t} \in (0,1)^p$, the partial derivative $\frac{\partial\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}}{\partial t_j}$ for each $j = 1,\ldots,p$ is equal to

$$L_{\boldsymbol{t}}^{-1}\left[E_j - E_j Z T_{\boldsymbol{t}} L_{\boldsymbol{t}}^{-1} T_{\boldsymbol{t}} - T_{\boldsymbol{t}} Z E_j L_{\boldsymbol{t}}^{-1} T_{\boldsymbol{t}}\right]\left(\frac{X^\top \boldsymbol{y}}{n}\right),$$

where $Z = n^{-1}\left(X^\top X - \delta I\right)$ and E_j is a square matrix of dimension $p\times p$ with 1 at the (j,j)th position and 0 everywhere else.

Proof of Lemma 1 Existence of $\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}$ for every $\boldsymbol{t} \in (0,1)^p$ and $\delta > 0$ follows from Theorem 1 which states that $L_{\boldsymbol{t}}$ is positive-definite and hence guarantees the invertibility of $L_{\boldsymbol{t}}$. Since $\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}} = L_{\boldsymbol{t}}^{-1}T_{\boldsymbol{t}}X^\top \boldsymbol{y}/n$, using matrix calculus, for any $j = 1,\ldots,p$,

$$\frac{\partial\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}}{\partial t_j} = \frac{\partial\left(L_{\boldsymbol{t}}^{-1}T_{\boldsymbol{t}}\right)}{\partial t_j}\left(\frac{X^\top \boldsymbol{y}}{n}\right)$$

$$= \left[\frac{\partial L_{\boldsymbol{t}}^{-1}}{\partial t_j} T_{\boldsymbol{t}} + L_{\boldsymbol{t}}^{-1} \frac{\partial T_{\boldsymbol{t}}}{\partial t_j}\right] \left(\frac{X^{\mathsf{T}} \boldsymbol{y}}{n}\right)$$
$$= \left[L_{\boldsymbol{t}}^{-1} \frac{\partial T_{\boldsymbol{t}}}{\partial t_j} - L_{\boldsymbol{t}}^{-1} \frac{\partial L_{\boldsymbol{t}}}{\partial t_j} L_{\boldsymbol{t}}^{-1} T_{\boldsymbol{t}}\right] \left(\frac{X^{\mathsf{T}} \boldsymbol{y}}{n}\right),$$

where we used differentiation of an invertible matrix which implies

$$\frac{\partial L_{\boldsymbol{t}}^{-1}}{\partial t_j} = -L_{\boldsymbol{t}}^{-1} \frac{\partial L_{\boldsymbol{t}}}{\partial t_j} L_{\boldsymbol{t}}^{-1}.$$

Since $\partial T_{\boldsymbol{t}}/\partial t_j = E_j$, and the fact that $L_{\boldsymbol{t}} = T_{\boldsymbol{t}} Z T_{\boldsymbol{t}} + \delta I/n$, we get

$$\frac{\partial L_{\boldsymbol{t}}}{\partial t_j} = \frac{\partial T_{\boldsymbol{t}}}{\partial t_i} Z T_{\boldsymbol{t}} + T_{\boldsymbol{t}} Z \frac{\partial T_{\boldsymbol{t}}}{\partial t_j}$$
$$= E_j Z T_{\boldsymbol{t}} + T_{\boldsymbol{t}} Z E_j.$$

Therefore, $L_{\boldsymbol{t}}^{-1} \frac{\partial T_{\boldsymbol{t}}}{\partial t_j} - L_{\boldsymbol{t}}^{-1} \frac{\partial L_{\boldsymbol{t}}}{\partial t_j} L_{\boldsymbol{t}}^{-1} T_{\boldsymbol{t}}$ is equal to

$$L_{\boldsymbol{t}}^{-1} E_j - L_{\boldsymbol{t}}^{-1} E_j Z T_{\boldsymbol{t}} L_{\boldsymbol{t}}^{-1} T_{\boldsymbol{t}} - L_{\boldsymbol{t}}^{-1} T_{\boldsymbol{t}} Z E_j L_{\boldsymbol{t}}^{-1} T_{\boldsymbol{t}}$$
$$= L_{\boldsymbol{t}}^{-1} \left[E_j - E_j Z T_{\boldsymbol{t}} L_{\boldsymbol{t}}^{-1} T_{\boldsymbol{t}} - T_{\boldsymbol{t}} Z E_j L_{\boldsymbol{t}}^{-1} T_{\boldsymbol{t}}\right].$$

This completes the proof Lemma 1. □

Proof of Theorem 4 To obtain the gradient $\nabla f_\lambda(\boldsymbol{t})$ for $\boldsymbol{t} \in (0,1)^p$, let $\boldsymbol{\gamma_t} = T_{\boldsymbol{t}} \widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}} = \boldsymbol{t} \odot \widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}$. Then,

$$\|\boldsymbol{y} - X_{\boldsymbol{t}} \widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}\|_2^2 = \|\boldsymbol{y} - X \boldsymbol{\gamma_t}\|_2^2$$
$$= \boldsymbol{y}^{\mathsf{T}} \boldsymbol{y} - 2 \boldsymbol{\gamma}_{\boldsymbol{t}}^{\mathsf{T}} \left(X^{\mathsf{T}} \boldsymbol{y}\right) + \boldsymbol{\gamma}_{\boldsymbol{t}}^{\mathsf{T}} \left(X^{\mathsf{T}} X\right) \boldsymbol{\gamma_t}. \tag{A12}$$

Consequently,

$$\frac{\partial f_\lambda(\boldsymbol{t})}{\partial t_j} = \frac{1}{n} \frac{\partial}{\partial t_j} \left[\|\boldsymbol{y} - X_{\boldsymbol{t}} \widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}\|_2^2\right] + \lambda$$
$$= -\frac{2}{n} \left(\frac{\partial \boldsymbol{\gamma_t}}{\partial t_j}\right)^{\mathsf{T}} \left(X^{\mathsf{T}} \boldsymbol{y}\right)$$
$$+ \frac{2}{n} \left(\frac{\partial \boldsymbol{\gamma_t}}{\partial t_j}\right)^{\mathsf{T}} \left(X^{\mathsf{T}} X\right) \boldsymbol{\gamma_t} + \lambda$$
$$= \frac{2}{n} \left(\frac{\partial \boldsymbol{\gamma_t}}{\partial t_j}\right)^{\mathsf{T}} \left[\left(X^{\mathsf{T}} X\right) \boldsymbol{\gamma_t} - \left(X^{\mathsf{T}} \boldsymbol{y}\right)\right] + \lambda$$
$$= 2 \left(\frac{\partial \boldsymbol{\gamma_t}}{\partial t_j}\right)^{\mathsf{T}} \boldsymbol{a_t} + \lambda, \tag{A13}$$

where $\boldsymbol{a_t} = n^{-1}[X^{\mathsf{T}} X \boldsymbol{\gamma_t} - X^{\mathsf{T}} \boldsymbol{y}]$. From the definitions of $\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}$ and $\boldsymbol{\gamma_t}$,

$$\frac{\partial \boldsymbol{\gamma_t}}{\partial t_j} = \frac{\partial T_{\boldsymbol{t}} \widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}}{\partial t_j}$$
$$= \frac{\partial T_{\boldsymbol{t}}}{\partial t_j} \widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}} + T_{\boldsymbol{t}} \frac{\partial \widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}}{\partial t_j},$$
$$= E_j \widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}} + T_{\boldsymbol{t}} L_{\boldsymbol{t}}^{-1} \left[E_j - E_j Z T_{\boldsymbol{t}} L_{\boldsymbol{t}}^{-1} T_{\boldsymbol{t}}\right.$$
$$\left. - T_{\boldsymbol{t}} Z E_j L_{\boldsymbol{t}}^{-1} T_{\boldsymbol{t}}\right] \left(\frac{X^{\mathsf{T}} \boldsymbol{y}}{n}\right),$$

which is obtained using Lemma 1 and the fact that $\partial T_{\boldsymbol{t}}/\partial t_j = E_j$ and $Z = n^{-1}\left(X^{\mathsf{T}} X - \delta I\right)$. This in-turn yields that $\frac{\partial \boldsymbol{\gamma_t}}{\partial t_j}$ is equal to

$$E_j \widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}} + T_{\boldsymbol{t}} L_{\boldsymbol{t}}^{-1} \left[E_j \left(\frac{X^{\mathsf{T}} \boldsymbol{y}}{n}\right) - E_j Z \boldsymbol{\gamma_t} - T_{\boldsymbol{t}} Z E_j \widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}\right]$$
$$= E_j \widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}} - T_{\boldsymbol{t}} L_{\boldsymbol{t}}^{-1} E_j \boldsymbol{b_t} - T_{\boldsymbol{t}} L_{\boldsymbol{t}}^{-1} T_{\boldsymbol{t}} Z E_j \widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}, \tag{A14}$$

where we recall that

$$\boldsymbol{b_t} = Z \boldsymbol{\gamma_t} - \left(\frac{X^{\mathsf{T}} \boldsymbol{y}}{n}\right) = \boldsymbol{a_t} - \frac{\delta}{n} \boldsymbol{\gamma_t}.$$

For a further simplification, recall that $\boldsymbol{c_t} = L_{\boldsymbol{t}}^{-1} (\boldsymbol{t} \odot \boldsymbol{a_t})$ and $\boldsymbol{d_t} = Z (\boldsymbol{t} \odot \boldsymbol{c_t})$. Then, from (A14), the matrix $\partial \boldsymbol{\gamma_t}/\partial \boldsymbol{t}$ of dimension $p \times p$, with jth column being $\partial \boldsymbol{\gamma_t}/\partial t_j$, can be expressed as

$$\frac{\partial \boldsymbol{\gamma_t}}{\partial \boldsymbol{t}} = \mathsf{Diag}\left(\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}\right) - T_{\boldsymbol{t}} L_{\boldsymbol{t}}^{-1} \mathsf{Diag}\left(\boldsymbol{b_t}\right)$$
$$- T_{\boldsymbol{t}} L_{\boldsymbol{t}}^{-1} T_{\boldsymbol{t}} Z \mathsf{Diag}\left(\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}\right). \tag{A15}$$

From (A13), with $\mathbf{1}$ representing a vector of all ones, $\nabla f_\lambda(\boldsymbol{t})$ can be expressed as

$$\nabla f_\lambda(\boldsymbol{t}) = 2\mathsf{Diag}\left(\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}\right) \boldsymbol{a_t} - 2\mathsf{Diag}\left(\boldsymbol{b_t}\right) L_{\boldsymbol{t}}^{-1} T_{\boldsymbol{t}} \boldsymbol{a_t}$$
$$- 2\mathsf{Diag}\left(\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}\right) Z T_{\boldsymbol{t}} L_{\boldsymbol{t}}^{-1} T_{\boldsymbol{t}} \boldsymbol{a_t} + \lambda \mathbf{1}$$
$$= 2\left(\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}} \odot \boldsymbol{a_t}\right) - 2\mathsf{Diag}\left(\boldsymbol{b_t}\right) \boldsymbol{c_t}$$
$$- 2\mathsf{Diag}\left(\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}\right) Z T_{\boldsymbol{t}} \boldsymbol{c_t} + \lambda \mathbf{1}$$
$$= 2\left(\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}} \odot \boldsymbol{a_t}\right) - 2\left(\boldsymbol{b_t} \odot \boldsymbol{c_t}\right) - 2\left(\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}} \odot \boldsymbol{d_t}\right) + \lambda \mathbf{1}$$
$$= 2\left(\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}} \odot \left(\boldsymbol{a_t} - \boldsymbol{d_t}\right)\right) - 2\left(\boldsymbol{b_t} \odot \boldsymbol{c_t}\right) + \lambda \mathbf{1}$$
$$= \boldsymbol{\zeta_t} + \lambda \mathbf{1},$$

where

$$\boldsymbol{\zeta_t} = 2\left(\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}} \odot \left(\boldsymbol{a_t} - \boldsymbol{d_t}\right)\right) - 2\left(\boldsymbol{b_t} \odot \boldsymbol{c_t}\right).$$

Finally, recall that $g_\lambda(\boldsymbol{w}) = f_\lambda\left(\boldsymbol{t}(\boldsymbol{w})\right)$, $\boldsymbol{w} \in \mathbb{R}^p$, where the map $t(w) = \mathbf{1} - \exp(-w \odot w)$ and

$$f_\lambda(\boldsymbol{t}) = \frac{1}{n} \|\boldsymbol{y} - X_{\boldsymbol{t}} \widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}\|_2^2 + \lambda \sum_{j=1}^{p} t_j.$$

Then, from the chain rule of differentiation, for each $j = 1, \ldots, p$,

$$\frac{\partial g_\lambda(\boldsymbol{w})}{\partial w_j} = \frac{\partial f_\lambda(\boldsymbol{t})}{\partial t_j} \left(2 w_j \exp(-w_j^2)\right).$$

Alternatively, in short,

$$\nabla g_\lambda(\boldsymbol{w}) = \nabla f_\lambda(\boldsymbol{t}) \odot \left(2\boldsymbol{w} \odot \exp(-\boldsymbol{w} \odot \boldsymbol{w})\right). \tag{A16}$$

□

Proof of Proposition 5 From Theorem 4,we obtain $\nabla f_\lambda(\boldsymbol{t})$ as follows,

$$\nabla f_\lambda(\boldsymbol{t}) = \zeta_{\boldsymbol{t}} + \lambda \mathbf{1},$$

where $\zeta_{\boldsymbol{t}} \in \mathbb{R}^p$, recalling from Theorem 4, is given by

$$\boldsymbol{\zeta_t} = 2\left(\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}} \odot (\boldsymbol{a_t} - \boldsymbol{d_t})\right) - 2\left(\boldsymbol{b_t} \odot \boldsymbol{c_t}\right), \qquad \text{(A17)}$$

with $\boldsymbol{a_t} = n^{-1}[X^{\mathsf{T}}X(\boldsymbol{t} \odot \widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}) - X^{\mathsf{T}}\boldsymbol{y}]$, $\boldsymbol{b_t} = \boldsymbol{a_t} - n^{-1}\delta(\boldsymbol{t} \odot \widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}})$, $\boldsymbol{c_t} = L_{\boldsymbol{t}}^{-1}(\boldsymbol{t} \odot \boldsymbol{a_t})$, and $\boldsymbol{d_t} = n^{-1}[X^{\mathsf{T}}X - \delta I](\boldsymbol{t} \odot \boldsymbol{c_t})$.

From the definition of $\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}$, we can show that for any j, if we fix t_i for all $i \neq j$, then the jth component of $\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}$ goes to zero as $t_j \downarrow 0$, that is, $\lim_{t_j \downarrow 0} \widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}}(j) = 0$. Similarly, $\lim_{t_j \downarrow 0} \boldsymbol{c_t}(j) = 0$. Therefore, from the expression of $\boldsymbol{\zeta_t}$ in (A17), we have $\lim_{t_j \downarrow 0} \boldsymbol{\zeta_t}(j) = 0$. □

For proving Theorem 6, we use Lemma 2 which is well-known as the Woodbury matrix identity or Duncan Inversion Formula; we refer to Woodbury (1950) for a proof of Lemma 2.

Lemma 2 For any conformable matrices A, B_1, and B_2, and C, the matrix $(A + B_1 C B_2)^{-1}$ is equal to

$$A^{-1} - A^{-1}B_1\left(C^{-1} + B_2 A^{-1} B_1\right)^{-1} B_2 A^{-1}.$$

Proof of Theorem 6 Recall the expression of $L_{\boldsymbol{t}}$:

$$L_{\boldsymbol{t}} = \frac{1}{n}\left[X_{\boldsymbol{t}}^{\mathsf{T}} X_{\boldsymbol{t}} + \delta\left(I - T_{\boldsymbol{t}}^2\right)\right].$$

From the definition, for $\boldsymbol{t} \in [0,1)^p$,

$$S_{\boldsymbol{t}} = \frac{n}{\delta}(I - T_{\boldsymbol{t}}^2)^{-1},$$

which exists. Further, if we take

$$A = \frac{\delta}{n}\left(I - T_{\boldsymbol{t}}^2\right), \quad B_1 = B_2^{\mathsf{T}} = \frac{1}{\sqrt{n}} X_{\boldsymbol{t}}^{\mathsf{T}}, \quad \text{and} \quad C = I,$$

in Lemma 2, then,

$$L_{\boldsymbol{t}}^{-1} = S_{\boldsymbol{t}} - \frac{1}{n} S_{\boldsymbol{t}} X_{\boldsymbol{t}}^{\mathsf{T}}\left(I + \frac{1}{n} X_{\boldsymbol{t}} S_{\boldsymbol{t}} X_{\boldsymbol{t}}^{\mathsf{T}}\right)^{-1} X_{\boldsymbol{t}} S_{\boldsymbol{t}}.$$

Since $S_{\boldsymbol{t}}$ is a diagonal matrix,

$$\widetilde{L}_{\boldsymbol{t}} = I + \frac{1}{n} X_{\boldsymbol{t}} S_{\boldsymbol{t}} X_{\boldsymbol{t}}^{\mathsf{T}}$$

is a symmetric positive-definite matrix of dimension $n \times n$. Thus, $\widetilde{L}_{\boldsymbol{t}}^{-1}$ exists and

$$L_{\boldsymbol{t}}^{-1} = S_{\boldsymbol{t}} - \frac{1}{n} S_{\boldsymbol{t}} X_{\boldsymbol{t}}^{\mathsf{T}} \widetilde{L}_{\boldsymbol{t}}^{-1} X_{\boldsymbol{t}} S_{\boldsymbol{t}}.$$

□

Proof of Theorem 7 For the same reasons mentioned in the proof of Theorem 2, without loss of generality we can assume that all the zero elements of $\boldsymbol{t}$ appear at the end of $\boldsymbol{t}$; that is, $\boldsymbol{t}$ is of the form:

$$\boldsymbol{t} = (t_1, \ldots, t_m, 0, \ldots, 0),$$

where m indicates the number of non-zero elements in $\boldsymbol{t}$. Then, $L_{\boldsymbol{t}}$ is given by

$$L_{\boldsymbol{t}} = \frac{1}{n}\left[\begin{pmatrix} \left(X_{\boldsymbol{t}}^{\mathsf{T}} X_{\boldsymbol{t}}\right)_\star & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} + \delta \begin{pmatrix} \left(I - \left(T_{\boldsymbol{t}}^2\right)_\star\right) & \mathbf{0} \\ \mathbf{0} & I \end{pmatrix}\right] \qquad \text{(A18)}$$

$$= \begin{pmatrix} (L_{\boldsymbol{t}})_\star & \mathbf{0} \\ \mathbf{0} & \frac{\delta}{n} I \end{pmatrix}.$$

Since $\boldsymbol{t} \in [0,1)^p$, from Theorem 1, $L_{\boldsymbol{t}}$ is a positive-definite matrix. Every principle submatrix of a positive-definite matrix is also positive-definite. Thus, $(L_{\boldsymbol{t}})_\star$ is positive-definite and hence invertible. Using Theorem 8 (ii),

$$L_{\boldsymbol{t}}^{-1} = \begin{pmatrix} \left((L_{\boldsymbol{t}})_\star\right)^{-1} & \mathbf{0} \\ \mathbf{0} & I \end{pmatrix} \begin{pmatrix} I & \mathbf{0} \\ \mathbf{0} & \frac{n}{\delta} I \end{pmatrix}$$

$$= \begin{pmatrix} \left((L_{\boldsymbol{t}})_\star\right)^{-1} & \mathbf{0} \\ \mathbf{0} & \frac{n}{\delta} I \end{pmatrix}. \qquad \text{(A19)}$$

Now observe that

$$\begin{pmatrix} \frac{X_{\boldsymbol{t}}^{\mathsf{T}} \boldsymbol{y}}{n} \end{pmatrix} = \begin{pmatrix} \left(\frac{X_{\boldsymbol{t}}^{\mathsf{T}} \boldsymbol{y}}{n}\right)_\star \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} (\boldsymbol{t})_\star \odot \left(\frac{X^{\mathsf{T}} \boldsymbol{y}}{n}\right)_\star \\ \mathbf{0} \end{pmatrix}.$$

Since $\widetilde{\boldsymbol{\beta}}_{\boldsymbol{t}} = L_{\boldsymbol{t}}^{-1} X_{\boldsymbol{t}}^{\mathsf{T}} \boldsymbol{y}/n$, using (A19), we establish the desired expressions for $(\widetilde{\boldsymbol{\beta}}_t)_0$ and $(\widetilde{\boldsymbol{\beta}}_t)_+$. Similar arguments yield the desired expressions for $(c_t)_0$ and $(c_t)_+$; hence for conciseness that proof is omitted here. □

References

Bertsimas D, King A, Mazumder R (2016) Best subset selection via a modern optimization lens. The Annals of Statistics 44(2):813 – 852

Castro-González N, Martínez-Serrano M, Robles J (2015) Expressions for the moore–penrose inverse of block matrices involving the schur complement. Linear Algebra and its Applications 471:353–368

Efroymson MA (1966) Stepwise regression—a backward and forward look. Presented at the Eastern Regional Meetings of of the Institute of Mathematical Statistics, Florham Park, New Jersey

Fan J, Li R (2006) Statistical challenges with high dimensionality: feature selection in knowledge discovery. In: International Congress of Mathematicians. Vol. III. Eur. Math. Soc., Zürich, p 595–622

Fan J, Lv J (2010) A selective overview of variable selection in high dimensional feature space. Statist Sinica 20(1):101–148

Furnival GM, Wilson RW (2000) Regressions by leaps and bounds. Technometrics 42:69–79

Golub GH, Van Loan CF (1996) Matrix computations, 3rd edn. Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD

Gurobi Optimization, limited liability company (2022) Gurobi Optimizer Reference Manual. URL https://www.gurobi.com

Hastie T, Tibshirani R, Tibshirani R (2018) bestsubset: Tools for best subset selection in regression. R package version 1.0.10

Hastie T, Tibshirani R, Tibshirani R (2020) Best subset, forward stepwise or lasso? analysis and recommendations based on extensive comparisons. Statistical Science 35(4):579–592

Hazimeh H, Mazumder R (2020a) Fast best subset selection: coordinate descent and local combinatorial optimization algorithms. Oper Res 68(5):1517–1537

Hazimeh H, Mazumder R (2020b) Fast best subset selection: Coordinate descent and local combinatorial optimization algorithms. Operations Research 68(5):1517–1537

Hazimeh H, Mazumder R, Nonet T (2023) L0Learn: Fast Algorithms for Best Subset Selection. R package version 2.1.0

Hocking RR, Leslie RN (1967) Selection of the best subset in regression analysis. Technometrics 9:531–540

Kochenderfer MJ, Wheeler TA (2019) Algorithms for optimization. MIT Press, Cambridge, MA

Mathur A, Moka S, Botev Z (2023) Column subset selection and Nyström approximation via continuous optimization. 2304.09678

Miller A (2019) Subset selection in regression, Monographs on Statistics and Applied Probability, vol 95. Chapman & Hall/CRC, Boca Raton, FL

Moka S, Liquet B, Zhu H, et al (2022) COMBSS: Best subset selection via continuous optimization. URL https://arxiv.org/abs/2205.02617

Müller S, Welsh AH (2010) On model selection curves. Intnl Statist Reviews 78(2):240–256

Natarajan BK (1995) Sparse approximate solutions to linear systems. SIAM J Comput 24(2):227–234

Tarr G, Muller S, Welsh AH (2018) mplot: An r package for graphical model stability and variable selection procedures. J Statist Software 83(9):1–28

Tian Y, Takane Y (2005) Schur complements and Banachiewicz-Schur forms. Electron J Linear Algebra 13:405–418

Tibshirani R (1996) Regression shrinkage and selection via the lasso. J Roy Statist Soc Ser B 58(1):267–288

Woodbury MA (1950) Inverting modified matrices. Princeton University, Princeton, N. J., statistical Research Group, Memo. Rep. no. 42,

Zhu J, Wen C, Zhu J, et al (2020) A polynomial algorithm for best-subset selection problem. Proc Natl Acad Sci USA 117(52):33,117–33,123