Branch-and-bound Algorithm for Optimal Sparse Canonical Correlation Analysis

Akihisa Watanabe[a], Ryuta Tamura[b,c], Yuichi Takano[d,*], Ryuhei Miyashiro[e]

[a] *Department of Industrial Engineering and Economics, Tokyo Institute of Technology, 2-12-1 Ookayama, Meguro-ku, 152-8552, Tokyo, Japan*
[b] *Graduate School of Engineering, Tokyo University of Agriculture and Technology, 2-24-16 Naka-cho, Koganei-shi, 184-8588, Tokyo, Japan*
[c] *October Sky Co., Ltd., Zelkova Bldg., 1-25-12 Fuchu-cho, Fuchu-shi, 183-0055, Tokyo, Japan*
[d] *Faculty of Engineering, Information and Systems, University of Tsukuba, 1-1-1 Tennodai, Tsukuba-shi, 305-8573, Ibaraki, Japan*
[e] *Institute of Engineering, Tokyo University of Agriculture and Technology, 2-24-16 Naka-cho, Koganei-shi, 184-8588, Tokyo, Japan*

Abstract

Canonical correlation analysis (CCA) is a family of multivariate statistical methods for extracting mutual information contained in multiple datasets. To improve the interpretability of CCA, here we focus on the mixed-integer optimization (MIO) approach to sparse estimation. This approach was first proposed for sparse linear regression in the 1970s, but it has recently received renewed attention due to advances in optimization algorithms and computer hardware. We first derive an MIO problem for optimal sparse CCA estimation. To solve this MIO problem exactly, we propose a branch-and-bound algorithm based on the generalized eigenvalue problem for computing effective lower and upper bounds. We prove that our algorithm finds a solution with guaranteed optimality in terms of the canonical correlation. Computational results demonstrate that our method is much faster than direct application of optimization software to the MIO problem. Moreover, our method can provide better-quality solutions than can forward stepwise selection and L_1-regularized estimation in terms of generalization performance. These results enhance the potential of optimal sparse estimation in multivariate statistical analyses.

*Corresponding author

Keywords: canonical correlation analysis, branch-and-bound algorithm, sparse estimation, mixed-integer optimization, statistics

1. Introduction

Canonical correlation analysis (CCA), first proposed by Hotelling in the 1930s [28, 29], is a family of multivariate statistical methods for extracting mutual information shared by two or more datasets. CCA computes linear combinations of variables in each dataset such that canonical correlation between the linear combinations is maximized. Notably, CCA includes, as special cases, linear regression, principal component analysis (PCA), and partial least squares (PLS) regression [67]. Many successful applications of CCA that uncover multivariate relationships among different measurements can be found in various scientific fields [54, 56, 67].

A number of prior studies have sought to improve the representational power of CCA. Several techniques for analyzing nonlinear relationships have been employed in CCA, including artificial neural networks [2, 30, 34, 59] and kernel methods [1, 4, 19, 25]. Another major area of research is related to enhancing the interpretability of CCA by means of sparse estimation, which aims at decreasing the number of nonzero weights constructing linear combinations. Sparse estimation enables us to identify significant subsets of variables used in linear combinations and thus brings a variety of analytical benefits [13, 23, 36, 38].

A direct method for optimal sparse estimation involves scanning all possible subsets of variables. However, such exhaustive search methods are often computationally infeasible because the number of possible subsets grows exponentially with respect to the number of candidate variables [40, 46]. In contrast, stepwise selection is a fast greedy heuristic for sparse estimation in which one variable at a time is repeatedly added and eliminated. Stepwise selection methods are commonly used for CCA [54, 60] but often provide suboptimal solutions. Recently, various regularization (or shrinkage) methods have been proposed for sparse CCA [14, 20, 24, 53, 55, 64] and applied to genomic data analyses [35, 37, 45, 58, 63]. However, these regularization methods produce biased estimates and sometimes yield low-quality solutions owing to an adverse effect of the regularization term.

This paper focuses on the mixed-integer optimization (MIO) approach to sparse estimation. This approach was first proposed for sparse linear regression in the 1970s [3], but it has recently received renewed attention due to

advances in optimization algorithms and computer hardware [8, 15, 26, 33, 39, 57]. In contrast to many sparse estimation algorithms, the MIO approach has the advantage of selecting an optimal subset of variables with respect to criterion functions, including Mallows' C_p [41], adjusted R^2 [42], information criteria [21, 42], mRMR [44], and a cross-validation criterion [50]. MIO-based sparse estimation methods have been extended to logistic regression [6, 49], ordinal regression [43, 48], count regression [47], and elimination of multicollinearity [7, 9, 51, 52].

Several studies have implemented algorithms specialized for exactly solving MIO problems for sparse estimation. Bertsimas and Shioda [10] described a branch-and-bound algorithm for cardinality-constrained quadratic optimization, with applications to sparse linear regression. Kimura and Waki [32] designed a branch-and-bound algorithm for minimizing the Akaike information criterion for linear regression, and Kimura [31] applied this algorithm to logistic regression. Hazimeh et al. [27] devised a branch-and-bound algorithm using specialized first-order methods for sparse linear regression. Dedieu et al. [17] proposed the integrality generation algorithm to speed up MIO computations for sparse classification. Recently, Berk and Bertsimas [5] devised a tailored branch-and-bound algorithm for optimal sparse PCA. This algorithm can be extended to sparse sufficient dimension reduction [66]. To our knowledge, however, no prior studies have developed a practicable algorithm for exactly solving the sparse CCA problem.

On the basis of the study [5] by Berk and Bertsimas on sparse PCA, we propose a high-performance branch-and-bound algorithm that computes an optimal solution to the sparse CCA problem. We first formulate the sparse CCA problem as an MIO problem. We next design our branch-and-bound algorithm for solving this MIO problem, where generalized eigenvalue problems are repeatedly solved to derive effective lower and upper bounds. We then prove that our algorithm gives a solution with guaranteed optimality in terms of the canonical correlation. Note that our algorithm can also be applied to sparse PCA and PLS regression problems, which are special cases of the sparse CCA problem.

We assess the efficacy of our method through computational experiments using real-world datasets downloaded from the UCI Machine Learning Repository [18]. Our method is much faster than direct application of optimization software to the MIO problem for sparse CCA estimation. Moreover, our method often provides better-quality solutions than do forward stepwise selection and L_1-regularized estimation in terms of generalization (out-of-

sample) performance.

Notation. Throughout this paper, we denote the set of consecutive integers ranging from 1 to n as

$$[n] := \begin{cases} \{1, 2, \ldots, n\} & \text{if } n \geq 1, \\ \emptyset & \text{otherwise.} \end{cases}$$

2. Canonical Correlation Analysis

We address the task of extracting mutual information from the following two vectors composed of random variables:

$$\boldsymbol{x} := (x_1, x_2, \ldots, x_p)^\top, \quad \boldsymbol{y} := (y_1, y_2, \ldots, y_q)^\top.$$

Suppose we are given a sample of n data instances, $(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathbb{R}^p \times \mathbb{R}^q$ for $i \in [n]$. For simplicity, we assume that all variables are centered based on the sample mean as

$$\mathrm{E}[\boldsymbol{x}] := \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i = \boldsymbol{0}, \quad \mathrm{E}[\boldsymbol{y}] := \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{y}_i = \boldsymbol{0}.$$

The sample covariance matrices are then expressed as

$$\boldsymbol{C}_{\boldsymbol{xx}} := \mathrm{E}[\boldsymbol{x}\boldsymbol{x}^\top] = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i\boldsymbol{x}_i^\top, \quad \boldsymbol{C}_{\boldsymbol{xy}} := \mathrm{E}[\boldsymbol{x}\boldsymbol{y}^\top] = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i\boldsymbol{y}_i^\top,$$
$$\boldsymbol{C}_{\boldsymbol{yx}} := \mathrm{E}[\boldsymbol{y}\boldsymbol{x}^\top] = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{y}_i\boldsymbol{x}_i^\top, \quad \boldsymbol{C}_{\boldsymbol{yy}} := \mathrm{E}[\boldsymbol{y}\boldsymbol{y}^\top] = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{y}_i\boldsymbol{y}_i^\top.$$

As shown in Figure 1, the *canonical variates* are constructed by the linear combinations of variables

$$u(\boldsymbol{x}) := \boldsymbol{a}^\top\boldsymbol{x} = a_1x_1 + a_2x_2 + \cdots + a_px_p, \tag{1}$$
$$v(\boldsymbol{y}) := \boldsymbol{b}^\top\boldsymbol{y} = b_1y_1 + b_2y_2 + \cdots + b_qy_q, , \tag{2}$$

where $\boldsymbol{a} := (a_1, a_2, \ldots, a_p)^\top$ and $\boldsymbol{b} := (b_1, b_2, \ldots, b_q)^\top$ are *(canonical) weight vectors* to be estimated. The *canonical correlation*, which quantifies the

amount of mutual information contained in the canonical variates, is then defined by the sample correlation between $u(\boldsymbol{x})$ and $v(\boldsymbol{y})$ as

$$\begin{aligned}\rho(u(\boldsymbol{x}), v(\boldsymbol{y})) &:= \frac{\mathrm{E}[u(\boldsymbol{x})v(\boldsymbol{y})]}{\sqrt{\mathrm{E}[u(\boldsymbol{x})^2]}\sqrt{\mathrm{E}[v(\boldsymbol{y})^2]}} \\ &= \frac{(\sum_{i=1}^n \boldsymbol{a}^\top \boldsymbol{x}_i \boldsymbol{y}_i^\top \boldsymbol{b})/n}{\sqrt{(\sum_{i=1}^n \boldsymbol{a}^\top \boldsymbol{x}_i \boldsymbol{x}_i^\top \boldsymbol{a})/n}\sqrt{(\sum_{i=1}^n \boldsymbol{b}^\top \boldsymbol{y}_i \boldsymbol{y}_i^\top \boldsymbol{b})/n}} \\ &= \frac{\boldsymbol{a}^\top \boldsymbol{C}_{\boldsymbol{xy}} \boldsymbol{b}}{\sqrt{\boldsymbol{a}^\top \boldsymbol{C}_{\boldsymbol{xx}} \boldsymbol{a}}\sqrt{\boldsymbol{b}^\top \boldsymbol{C}_{\boldsymbol{yy}} \boldsymbol{b}}}. \end{aligned} \tag{3}$$

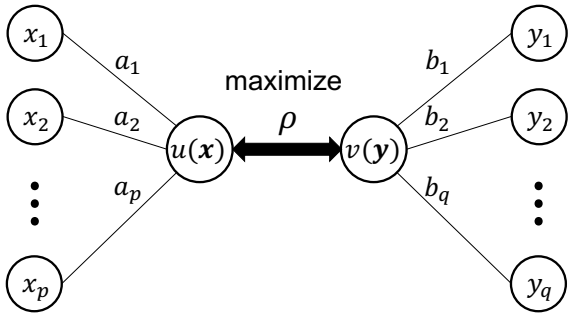

Figure 1: Conceptual diagram of canonical correlation analysis.

CCA estimates weight vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ such that the canonical correlation (3) is maximized as

$$\text{maximize} \quad \boldsymbol{a}^\top \boldsymbol{C}_{\boldsymbol{xy}} \boldsymbol{b} \tag{4}$$

$$\text{subject to} \quad \boldsymbol{a}^\top \boldsymbol{C}_{\boldsymbol{xx}} \boldsymbol{a} = \boldsymbol{b}^\top \boldsymbol{C}_{\boldsymbol{yy}} \boldsymbol{b} = 1, \tag{5}$$

$$\boldsymbol{a} \in \mathbb{R}^p, \ \boldsymbol{b} \in \mathbb{R}^q. \tag{6}$$

To deal with the constrained optimization problem (4)–(6), we introduce the Lagrangian function

$$\mathcal{L}(\boldsymbol{a}, \boldsymbol{b}, \lambda_{\boldsymbol{a}}, \lambda_{\boldsymbol{b}}) := \boldsymbol{a}^\top \boldsymbol{C}_{\boldsymbol{xy}} \boldsymbol{b} - \lambda_{\boldsymbol{a}}(\boldsymbol{a}^\top \boldsymbol{C}_{\boldsymbol{xx}} \boldsymbol{a} - 1) - \lambda_{\boldsymbol{b}}(\boldsymbol{b}^\top \boldsymbol{C}_{\boldsymbol{yy}} \boldsymbol{b} - 1),$$

where $\lambda_{\boldsymbol{a}}$ and $\lambda_{\boldsymbol{b}}$ are Lagrange multipliers. The optimality conditions are expressed as

$$\nabla_{\boldsymbol{a}} \mathcal{L}(\boldsymbol{a}, \boldsymbol{b}, \lambda_{\boldsymbol{a}}, \lambda_{\boldsymbol{b}}) = \boldsymbol{C}_{\boldsymbol{xy}} \boldsymbol{b} - 2\lambda_{\boldsymbol{a}} \boldsymbol{C}_{\boldsymbol{xx}} \boldsymbol{a} = \boldsymbol{0}, \tag{7}$$

$$\nabla_{\boldsymbol{b}} \mathcal{L}(\boldsymbol{a}, \boldsymbol{b}, \lambda_{\boldsymbol{a}}, \lambda_{\boldsymbol{b}}) = \boldsymbol{C}_{\boldsymbol{yx}} \boldsymbol{a} - 2\lambda_{\boldsymbol{b}} \boldsymbol{C}_{\boldsymbol{yy}} \boldsymbol{b} = \boldsymbol{0}. \tag{8}$$

We multiply Eqs. (7) and (8) by $\boldsymbol{a}^\top$ and $\boldsymbol{b}^\top$, respectively, from the left. It follows from Eq. (5) that

$$\boldsymbol{a}^\top \boldsymbol{C}_{\boldsymbol{xy}} \boldsymbol{b} = 2\lambda_{\boldsymbol{a}} \boldsymbol{a}^\top \boldsymbol{C}_{\boldsymbol{xx}} \boldsymbol{a} = 2\lambda_{\boldsymbol{a}}, \tag{9}$$

$$\boldsymbol{b}^\top \boldsymbol{C}_{\boldsymbol{yx}} \boldsymbol{a} = 2\lambda_{\boldsymbol{b}} \boldsymbol{b}^\top \boldsymbol{C}_{\boldsymbol{yy}} \boldsymbol{b} = 2\lambda_{\boldsymbol{b}}. \tag{10}$$

Since the left-hand sides of Eqs. (9) and (10) are equal after transposition, we can substitute $\lambda = 2\lambda_{\boldsymbol{a}} = 2\lambda_{\boldsymbol{b}}$ into the optimality conditions (7) and (8). Therefore, the optimality condition of problem (4)–(6) is posed as the following *generalized eigenvalue problem*:

$$\begin{bmatrix} \boldsymbol{O} & \boldsymbol{C}_{\boldsymbol{xy}} \\ \boldsymbol{C}_{\boldsymbol{yx}} & \boldsymbol{O} \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix} = \lambda \begin{bmatrix} \boldsymbol{C}_{\boldsymbol{xx}} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{C}_{\boldsymbol{yy}} \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix}. \tag{11}$$

Note that its eigenvalue λ coincides with the objective function (4) through Eq. (9). Consequently, the canonical correlation (3) is maximized by the eigenvector $(\boldsymbol{a}, \boldsymbol{b})$ corresponding to the maximum eigenvalue $\lambda^*_{\max}$ of the generalized eigenvalue problem (11).

When covariance matrices $\boldsymbol{C}_{\boldsymbol{xx}}$ and $\boldsymbol{C}_{\boldsymbol{yy}}$ are invertible, problem (11) can be reduced to a standard eigenvalue problem. It follows from Eq. (7) that

$$\boldsymbol{a} = \frac{\boldsymbol{C}_{\boldsymbol{xx}}^{-1} \boldsymbol{C}_{\boldsymbol{xy}} \boldsymbol{b}}{\lambda}. \tag{12}$$

Substituting Eq. (12) into Eq. (8) yields the following *standard eigenvalue problem*:

$$\boldsymbol{C}_{\boldsymbol{yy}}^{-1} \boldsymbol{C}_{\boldsymbol{yx}} \boldsymbol{C}_{\boldsymbol{xx}}^{-1} \boldsymbol{C}_{\boldsymbol{xy}} \boldsymbol{b} = \lambda^2 \boldsymbol{b}. \tag{13}$$

After its eigenvector $\boldsymbol{b}$ and eigenvalue λ^2 are computed, the weight vector $\boldsymbol{a}$ is obtained from Eq. (12).

3. Mixed-integer Optimization Formulation

Let $\boldsymbol{z} := (z_1, z_2, \ldots, z_{p+q})^\top$ be a vector composed of binary decision variables for subset selection, namely,

$$z_j = \begin{cases} 1 & \text{if } x_j \text{ is selected (i.e., } a_j \neq 0), \\ 0 & \text{otherwise} \end{cases} \quad (j \in [p]),$$

$$z_{p+j} = \begin{cases} 1 & \text{if } y_j \text{ is selected (i.e., } b_j \neq 0), \\ 0 & \text{otherwise} \end{cases} \quad (j \in [q]).$$

We also introduce user-defined parameters $\theta_{\boldsymbol{x}} \in [p]$ and $\theta_{\boldsymbol{y}} \in [q]$ for specifying subset sizes.

The sparse CCA is formulated as the following MIO problem:

$$\begin{aligned}
\text{maximize} \quad & \boldsymbol{a}^\top \boldsymbol{C}_{\boldsymbol{xy}} \boldsymbol{b} && (14)\\
\text{subject to} \quad & \boldsymbol{a}^\top \boldsymbol{C}_{\boldsymbol{xx}} \boldsymbol{a} = \boldsymbol{b}^\top \boldsymbol{C}_{\boldsymbol{yy}} \boldsymbol{b} = 1, && (15)\\
& -M\boldsymbol{z} \leq \begin{pmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{pmatrix} \leq M\boldsymbol{z}, && (16)\\
& \sum_{j=1}^{p} z_j = \theta_{\boldsymbol{x}}, \quad \sum_{j=1}^{q} z_{p+j} = \theta_{\boldsymbol{y}}, && (17)\\
& \boldsymbol{a} \in \mathbb{R}^p,\ \boldsymbol{b} \in \mathbb{R}^q,\ \boldsymbol{z} \in \{0,1\}^{p+q}, && (18)
\end{aligned}$$

where M is a sufficiently large positive constant. If $z_j = 0$, then the corresponding variable is eliminated from the canonical variates (1) and (2) because its weight must be zero by Eq. (16). The number of nonzero weights is limited by Eq. (17). Note that Eq. (18) lists all decision variables.

It is crucial to estimate a reasonable value for M in Eq. (16) because an overly large M can cause numerical instabilities in MIO computations [61]. Let $\lambda_{\min}(\,\cdot\,)$ and $\lambda_{\max}(\,\cdot\,)$ respectively denote the minimum and maximum eigenvalues of a matrix. The following theorem states that M can be set to

$$\hat{M} := \max\left\{ \frac{1}{\sqrt{\lambda_{\min}(\boldsymbol{C}_{\boldsymbol{xx}})}},\ \frac{1}{\sqrt{\lambda_{\min}(\boldsymbol{C}_{\boldsymbol{yy}})}} \right\}.$$

Theorem 1. Assume that the covariance matrices $\boldsymbol{C}_{\boldsymbol{xx}}$ and $\boldsymbol{C}_{\boldsymbol{yy}}$ are positive definite. If $(\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{R}^p \times \mathbb{R}^q$ satisfies Eq. (15), it also satisfies

$$-\hat{M} \leq \begin{pmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{pmatrix} \leq \hat{M}.$$

Proof. When Eq. (15) is satisfied, we have

$$|a_j| \leq \|\boldsymbol{a}\|_2 \leq \sqrt{\lambda_{\max}(\boldsymbol{C}_{\boldsymbol{xx}}^{-1})} = \frac{1}{\sqrt{\lambda_{\min}(\boldsymbol{C}_{\boldsymbol{xx}})}} \leq \hat{M} \quad (j \in [p]),$$

where the second inequality follows the properties of the ellipsoid $\{\boldsymbol{a} \in \mathbb{R}^p \mid \boldsymbol{a}^\top \boldsymbol{C}_{\boldsymbol{xx}} \boldsymbol{a} \leq 1\}$ (see, e.g., Section 2.2.2 of Boyd and Vandenberghe [12]). Similarly, we have $|b_j| \leq \hat{M}$ for all $j \in [q]$. □

4. Branch-and-bound Algorithm

This section presents our branch-and-bound algorithm for solving the MIO problem (14)–(18). Our algorithm is developed by extending the branch-and-bound algorithm [5] to sparse CCA estimation.

We first introduce the branching process of the algorithm (Section 4.1). We next define terminal nodes (Section 4.2) and then derive lower and upper bounds (Section 4.3) for bounding operations. Section 4.4 summarizes our branch-and-bound algorithm.

4.1. Branching Process

The branch-and-bound algorithm works on an *enumeration tree*, which is a binary search tree for systematic enumeration of candidate solutions $\boldsymbol{z} \in \{0,1\}^{p+q}$. As Figure 2 shows, each node of the enumeration tree is specified by a pair of binary vectors,

$$\boldsymbol{\ell} := (\ell_1, \ell_2, \ldots, \ell_{p+q})^\top \in \{0,1\}^{p+q}, \quad \boldsymbol{u} := (u_1, u_2, \ldots, u_{p+q})^\top \in \{0,1\}^{p+q}.$$

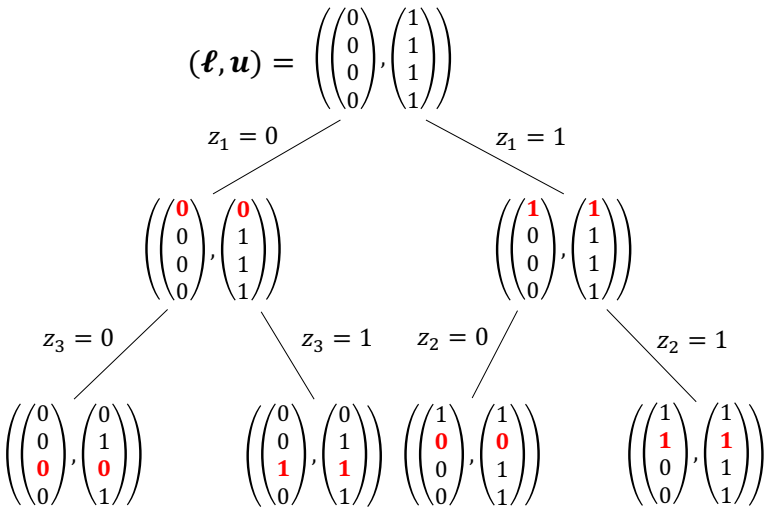

Figure 2: An enumeration tree in a branching process ($p + q = 4$).

For each node $(\boldsymbol{\ell}, \boldsymbol{u})$, a *subproblem* is posed by imposing the constraint

$\boldsymbol{\ell} \le \boldsymbol{z} \le \boldsymbol{u}$ on the MIO problem (14)–(18) as

$$\begin{aligned}
\mathrm{SP}(\boldsymbol{\ell}, \boldsymbol{u})\text{: maximize} \quad & \boldsymbol{a}^\top \boldsymbol{C}_{\boldsymbol{xy}} \boldsymbol{b} && (19)\\
\text{subject to} \quad & \boldsymbol{a}^\top \boldsymbol{C}_{\boldsymbol{xx}} \boldsymbol{a} = \boldsymbol{b}^\top \boldsymbol{C}_{\boldsymbol{yy}} \boldsymbol{b} = 1, && (20)\\
& -M\boldsymbol{z} \le \begin{pmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{pmatrix} \le M\boldsymbol{z}, && (21)\\
& \sum_{j=1}^{p} z_j = \theta_{\boldsymbol{x}}, \quad \sum_{j=1}^{q} z_{p+j} = \theta_{\boldsymbol{y}}, && (22)\\
& \boldsymbol{\ell} \le \boldsymbol{z} \le \boldsymbol{u}, && (23)\\
& \boldsymbol{a} \in \mathbb{R}^p,\ \boldsymbol{b} \in \mathbb{R}^q,\ \boldsymbol{z} \in \{0,1\}^{p+q}. && (24)
\end{aligned}$$

Note that the subproblem $\mathrm{SP}(\boldsymbol{\ell}, \boldsymbol{u})$ at the root node $(\boldsymbol{\ell}, \boldsymbol{u}) = (\boldsymbol{0}, \boldsymbol{1})$ coincides with the original problem (14)–(18).

We gradually fix $\boldsymbol{z}$ by using constraint (23) throughout the branching process (Figure 2). Specifically, setting $\ell_j = u_j = 0$ and $\ell_j = u_j = 1$ amounts to fixing $z_j = 0$ and $z_j = 1$, respectively. In particular, $\boldsymbol{z}$ is uniquely determined when $\boldsymbol{\ell} = \boldsymbol{u}$. Equations (22) and (23) also imply that the subproblem $\mathrm{SP}(\boldsymbol{\ell}, \boldsymbol{u})$ is feasible only when the following conditions are fulfilled:

$$\sum_{j=1}^{p} \ell_j \le \theta_{\boldsymbol{x}} \le \sum_{j=1}^{p} u_j, \quad \sum_{j=1}^{q} \ell_{p+j} \le \theta_{\boldsymbol{y}} \le \sum_{j=1}^{q} u_{p+j}. \tag{25}$$

4.2. Terminal Node

A *terminal node* is a node at which we no longer need to explore subsequent nodes. Terminal nodes are characterized by the *terminal-node function*

$$terminal((\boldsymbol{\ell}, \boldsymbol{u}), (s,t), \theta) := \begin{cases} true & \text{if } \sum_{j=s}^{t} \ell_j = \theta, \\ true & \text{if } \sum_{j=s}^{t} u_j = \theta, \\ false & \text{otherwise.} \end{cases}$$

In particular, note that

$$\begin{aligned}
& terminal((\boldsymbol{\ell}, \boldsymbol{u}), (1,p), \theta_{\boldsymbol{x}}) = true && (26)\\
\Longleftrightarrow\ & \sum_{j=1}^{p} \ell_j = \theta_{\boldsymbol{x}} \quad \text{or} \quad \sum_{j=1}^{p} u_j = \theta_{\boldsymbol{x}},
\end{aligned}$$

and that

$$terminal((\boldsymbol{\ell}, \boldsymbol{u}), (p+1, p+q), \theta_{\boldsymbol{y}}) = true \tag{27}$$
$$\iff \sum_{j=1}^{q} \ell_{p+j} = \theta_{\boldsymbol{y}} \quad \text{or} \quad \sum_{j=1}^{q} u_{p+j} = \theta_{\boldsymbol{y}}.$$

If both of conditions (26) and (27) are fulfilled, the corresponding node $(\boldsymbol{\ell}, \boldsymbol{u})$ is a terminal node because $\boldsymbol{z}$ is uniquely determined through Eqs. (22) and (23).

Suppose that $(\boldsymbol{\ell}, \boldsymbol{u})$ is a terminal node (i.e., $\boldsymbol{z}$ is fixed). As in Eq. (11), the subproblem $\mathrm{SP}(\boldsymbol{\ell}, \boldsymbol{u})$ is then reduced to the following generalized eigenvalue problem:

$$\begin{bmatrix} \boldsymbol{O} & \boldsymbol{C}_{\boldsymbol{x}(\boldsymbol{z})\boldsymbol{y}(\boldsymbol{z})} \\ \boldsymbol{C}_{\boldsymbol{y}(\boldsymbol{z})\boldsymbol{x}(\boldsymbol{z})} & \boldsymbol{O} \end{bmatrix} \begin{bmatrix} \boldsymbol{a}(\boldsymbol{z}) \\ \boldsymbol{b}(\boldsymbol{z}) \end{bmatrix} = \lambda(\boldsymbol{z}) \begin{bmatrix} \boldsymbol{C}_{\boldsymbol{x}(\boldsymbol{z})\boldsymbol{x}(\boldsymbol{z})} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{C}_{\boldsymbol{y}(\boldsymbol{z})\boldsymbol{y}(\boldsymbol{z})} \end{bmatrix} \begin{bmatrix} \boldsymbol{a}(\boldsymbol{z}) \\ \boldsymbol{b}(\boldsymbol{z}) \end{bmatrix}, \tag{28}$$

where

$$\boldsymbol{a}(\boldsymbol{z}) := (a_j \mid j \in [p],\ z_j = 1), \quad \boldsymbol{b}(\boldsymbol{z}) := (b_j \mid j \in [q],\ z_{p+j} = 1),$$
$$\boldsymbol{x}(\boldsymbol{z}) := (x_j \mid j \in [p],\ z_j = 1), \quad \boldsymbol{y}(\boldsymbol{z}) := (y_j \mid j \in [q],\ z_{p+j} = 1).$$

Let $\lambda^*_{\max}(\boldsymbol{z})$ and $(\boldsymbol{a}^*(\boldsymbol{z}), \boldsymbol{b}^*(\boldsymbol{z}))$ denote the maximum eigenvalue and the corresponding eigenvector of problem (28). For the fixed $\boldsymbol{z}$ at a terminal node $(\boldsymbol{\ell}, \boldsymbol{u})$, we can produce an optimal solution $(\boldsymbol{a}^{\mathrm{SP}}, \boldsymbol{b}^{\mathrm{SP}}, \boldsymbol{z}^{\mathrm{SP}})$ to the subproblem $\mathrm{SP}(\boldsymbol{\ell}, \boldsymbol{u})$ as

$$a_j^{\mathrm{SP}} := \begin{cases} a_j^*(\boldsymbol{z}) & \text{if } z_j = 1, \\ 0 & \text{otherwise} \end{cases} \quad (j \in [p]), \tag{29}$$

$$b_j^{\mathrm{SP}} := \begin{cases} b_j^*(\boldsymbol{z}) & \text{if } z_{p+j} = 1, \\ 0 & \text{otherwise} \end{cases} \quad (j \in [q]), \tag{30}$$

$$z_j^{\mathrm{SP}} := z_j \quad (j \in [p+q]). \tag{31}$$

4.3. Lower and Upper Bounds

When subproblem $\mathrm{SP}(\boldsymbol{\ell}, \boldsymbol{u})$ is feasible (i.e., Eq. (25) is satisfied), we compute lower and upper bounds on its optimal objective value.

To compute an upper bound on subproblem $\mathrm{SP}(\boldsymbol{\ell}, \boldsymbol{u})$, we consider the *relaxed subproblem* $\mathrm{RSP}(\boldsymbol{\ell}, \boldsymbol{u})$, where the subset size constraint (22) is removed from $\mathrm{SP}(\boldsymbol{\ell}, \boldsymbol{u})$. It is clear from Eq. (21) that the relaxed subproblem

RSP$(\boldsymbol{\ell}, \boldsymbol{u})$ has an optimal solution such that $\boldsymbol{z} = \boldsymbol{u}$ holds. Therefore, the generalized eigenvalue problem (28) with $\boldsymbol{z} = \boldsymbol{u}$ yields an optimal solution $(\boldsymbol{a}^{\mathrm{UB}}, \boldsymbol{b}^{\mathrm{UB}}, \boldsymbol{z}^{\mathrm{UB}})$ to the relaxed subproblem RSP$(\boldsymbol{\ell}, \boldsymbol{u})$, as in Eqs. (29)–(31). An upper bound on subproblem SP$(\boldsymbol{\ell}, \boldsymbol{u})$ is then

$$\mathit{upper}((\boldsymbol{\ell}, \boldsymbol{u})) = \lambda^*_{\max}(\boldsymbol{u}). \tag{32}$$

We next compute a lower bound on subproblem SP$(\boldsymbol{\ell}, \boldsymbol{u})$. To do so, we convert the relaxed solution $(\boldsymbol{a}^{\mathrm{UB}}, \boldsymbol{b}^{\mathrm{UB}}, \boldsymbol{z}^{\mathrm{UB}})$ into a solution $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{z})$ that is feasible for the subproblem SP$(\boldsymbol{\ell}, \boldsymbol{u})$. To accomplish this, we start with $\boldsymbol{z} = \boldsymbol{\ell}$ and increase it according to the following procedure: set $z_j = 1$ in descending order of $|a_j^{\mathrm{UB}}|$ for $j \in [p]$ with $(\ell_j, u_j) = (0, 1)$, and also set $z_{p+j} = 1$ in descending order of $|b_j^{\mathrm{UB}}|$ for $j \in [q]$ with $(\ell_{p+j}, u_{p+j}) = (0, 1)$. This procedure continues until Eq. (22) holds.

Let $\boldsymbol{z}^{\mathrm{LB}}$ denote a solution given by this procedure. We then solve the generalized eigenvalue problem (28) with $\boldsymbol{z} = \boldsymbol{z}^{\mathrm{LB}}$ to compute $(\boldsymbol{a}^{\mathrm{LB}}, \boldsymbol{b}^{\mathrm{LB}})$ as in Eqs. (29) and (30). A lower bound on subproblem SP$(\boldsymbol{\ell}, \boldsymbol{u})$ is then calculated as

$$\mathit{lower}((\boldsymbol{\ell}, \boldsymbol{u})) = \lambda^*_{\max}(\boldsymbol{z}^{\mathrm{LB}}). \tag{33}$$

4.4. Algorithm

Let f^* be the optimal objective value of the MIO problem (14)–(18). Then, $(\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{z}})$ is called an *ε-optimal solution* to problem (14)–(18) if it is feasible for problem (14)–(18) and satisfies

$$\hat{\boldsymbol{a}}^\top \boldsymbol{C}_{\boldsymbol{xy}} \hat{\boldsymbol{b}} \geq f^* - \varepsilon,$$

where $\varepsilon \geq 0$ is a user-defined tolerance parameter for optimality.

Algorithm 1 shows our branch-and-bound algorithm, which starts with initialization (line 1). Specifically, we add the root node $(\boldsymbol{\ell}, \boldsymbol{u}) = (\boldsymbol{0}, \boldsymbol{1})$ to the node set $\mathcal{N}$. We also set incumbent solution $(\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{z}})$, lower bound f^{LB}, and upper bound f^{UB}, where $\lambda^*_{\max}$ is given by the generalized eigenvalue problem (11).

We next select a node $(\boldsymbol{\ell}, \boldsymbol{u}) \in \mathcal{N}$ and check conditions (26) and (27) of terminal nodes (lines 3–7). If $(\boldsymbol{\ell}, \boldsymbol{u})$ is a terminal node (i.e., $k = 0$), then Eqs. (29)–(31) (lines 8–9) can be used to compute an optimal solution $(\boldsymbol{a}^{\mathrm{SP}}, \boldsymbol{b}^{\mathrm{SP}}, \boldsymbol{z}^{\mathrm{SP}})$ to the subproblem. If necessary, we update the lower bound and incumbent solution (lines 10–11).

If $(\boldsymbol{\ell}, \boldsymbol{u})$ is not a terminal node (i.e., $k \neq 0$), we explore new nodes by fixing $z_k \in \{0, 1\}$ (lines 12–14). We then compute the upper bound (32) and lower bound (33) on the new subproblems (lines 15–19). As Figure 2 shows, the upper-bound solution (i.e., $\boldsymbol{z} = \boldsymbol{u}$) of the new node is equal to that of its parent node when $z_k = 1$ (line 18). If necessary, we update the lower bound and incumbent solution, and add the new node to the node set (lines 20–23).

We remove from the node set all unpromising nodes that can be proved not to produce an optimal solution, and then calculate the upper bound (lines 24–25). The algorithm terminates if the optimality gap is sufficiently small (line 2).

The following theorem verifies the validity of our branch-and-bound algorithm.

Theorem 2. Algorithm 1 terminates in a finite number of iterations, then outputs an ε-optimal solution to problem (14)–(18).

Proof. Algorithm 1 terminates in a finite number of iterations because the number of all possible nodes is $2^{p+q+1} - 1$. Algorithm 1 also outputs an ε-optimal solution because its procedure for pruning the enumeration tree is valid (see, e.g., Proposition 1.3, Section II.4.1 of Wolsey and Nemhauser [65]). □

Remark 1. Let $(\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{z}})$ be a solution to problem (14)–(18). To compute the second pair of canonical weight vectors, we must solve problem (14)–(18) again after updating the covariance matrices based on orthogonal projections as

$$\begin{aligned}
\boldsymbol{C}_{\boldsymbol{xx}} &\leftarrow (\boldsymbol{I} - \hat{\boldsymbol{a}}\hat{\boldsymbol{a}}^\top \boldsymbol{C}_{\boldsymbol{xx}})^\top \boldsymbol{C}_{\boldsymbol{xx}} (\boldsymbol{I} - \hat{\boldsymbol{a}}\hat{\boldsymbol{a}}^\top \boldsymbol{C}_{\boldsymbol{xx}}), \\
\boldsymbol{C}_{\boldsymbol{xy}} &\leftarrow (\boldsymbol{I} - \hat{\boldsymbol{a}}\hat{\boldsymbol{a}}^\top \boldsymbol{C}_{\boldsymbol{xx}})^\top \boldsymbol{C}_{\boldsymbol{xy}} (\boldsymbol{I} - \hat{\boldsymbol{b}}\hat{\boldsymbol{b}}^\top \boldsymbol{C}_{\boldsymbol{yy}}), \\
\boldsymbol{C}_{\boldsymbol{yy}} &\leftarrow (\boldsymbol{I} - \hat{\boldsymbol{b}}\hat{\boldsymbol{b}}^\top \boldsymbol{C}_{\boldsymbol{yy}})^\top \boldsymbol{C}_{\boldsymbol{yy}} (\boldsymbol{I} - \hat{\boldsymbol{b}}\hat{\boldsymbol{b}}^\top \boldsymbol{C}_{\boldsymbol{yy}}).
\end{aligned}$$

We can repeat this process to compute subsequent pairs of canonical weight vectors [62].

5. Computational Results

This section evaluates the efficacy of our method through computational experiments. After explaining our experimental design (Section 5.1), we

Table 1: List of Datasets

Abbreviation	n	p	q	Original dataset [18]
Wine-R	1,599	6	5	Wine Quality (red wine)
Wine-W	4,898	6	5	Wine Quality (white wine)
Stdnt-M	395	13	13	Student Performance (mathematics)
Stdnt-P	649	13	13	Student Performance (Portuguese language)
Music	1,059	34	34	Geographical Original of Music
YearMSD	515,345	45	45	YearPredictionMSD

investigate the computational efficiency of the two eigenvalue problems (Section 5.2) and that of the node selection and branching rules (Section 5.3). We then examine the performance of our branch-and-bound algorithm by comparison with other sparse estimation methods (Section 5.4).

5.1. Experimental Design

Table 1 lists datasets downloaded from the UCI Machine Learning Repository [18]. Each categorical variable with two categories was treated as a dummy variable, and those with three or more categories were eliminated. Resultant variables were divided into the first p variables (i.e., $\boldsymbol{x}$) and the remaining q variables (i.e., $\boldsymbol{y}$). Each dataset consisting of n data instances was split 70 : 30 into training and test sets, with the training set used for sparse CCA estimation and the generalization (out-of-sample) performance estimated from applying the trained CCA model to the test set.

We implemented our branch-and-bound algorithm (Algorithm 1) in the Python programming language on Google Colaboratory [11], solving eigenvalue problems using the `scipy.linalg` module. The algorithm was terminated if it did not complete within 1,000 s. In those cases, the best feasible solution obtained within 1,000 s was taken as the result.

The following column labels are used in Tables 2–4:

LB: Lower bound f^{LB} on problem (14)–(18)

Gap: Optimality gap defined by $(f^{\mathrm{UB}} - f^{\mathrm{LB}})/f^{\mathrm{LB}}$

#Nodes: Number of nodes explored in Algorithm 1

Time: Computation time in seconds

The smallest optimality gaps for each problem instance are shown in bold.

5.2. Comparison of Eigenvalue Problems

As mentioned in Section 2, the CCA computation (4)–(6) can be reduced to the generalized eigenvalue problem (11) or the standard eigenvalue problem (13). Table 2 shows computational results from Algorithm 1, where the lower and upper bounds on subproblems were computed solving the following eigenvalue problems:

GenEgv: Algorithm 1 using the generalized eigenvalue problem (11),

StdEgv: Algorithm 1 using the standard eigenvalue problem (13).

Here, we adopted the depth-first search (DFS) node selection and WgtUB branching rules (see Section 5.3 for details). The results for the Wine-R and Wine-W datasets are omitted because the algorithm finished in very short times.

Table 2 shows that the StdEgv method was always faster than the GenEgv method when both finished within 1,000 s. Even when both methods failed to finish within 1,000 s, the results obtained by each were comparable. Note that the matrix size is $(p+q) \times (p+q)$ in the generalized eigenvalue problem (11), whereas it is $q \times q$ in the standard eigenvalue problem (13). The following sections therefore show computational results from Algorithm 1 using the standard eigenvalue problem.

5.3. Comparison of Node Selection and Branching Rules

The following node selection rules are commonly used in branch-and-bound algorithms (see, e.g., Section II.4.2 of Wolsey and Nemhauser [65]):

BFS: Breadth-first search

DFS: Depth-first search

LUB: Selection of the node having the largest upper bound

We used the following rules to select branching variables:

Random: Random selection of branching variables

WgtFull: Descending order of absolute values of canonical weights $(\boldsymbol{a}, \boldsymbol{b})$ given by the full model (4)–(6)

WgtUB: Largest absolute value of canonical weights $(\boldsymbol{a}^{\mathrm{UB}}, \boldsymbol{b}^{\mathrm{UB}})$ based on the upper bound f^{UB}

Tables 3 and 4 show computational results from Algorithm 1 with various combinations of node selection and branching rules (see also lines 3, 5, and 7 of Algorithm 1). Results for the Wine-R and Wine-W datasets are omitted because the algorithm finished in very short times.

Table 3 lists the results for small datasets (i.e., $(p, q) \leq (13, 13)$). The DFS rule was often the fastest node selection rule. As for branching rules, the Random rule was clearly worst, and the WgtUB rule was often the best in terms of both short computation time and small number of explored nodes.

Table 4 lists the results for large datasets (i.e., $(p, q) \geq (34, 34)$). Algorithm 1 was terminated due to the time limit in most problem instances. However, its computation using the DFS and WgtUB rules finished within 1,000 s for $(\theta_{\boldsymbol{x}}, \theta_{\boldsymbol{y}}) = (3, 3)$. In the following sections, we therefore show computational results from Algorithm 1 using the combination of DFS and WgtUB rules.

5.4. Comparison of Sparse Estimation Methods

We compare the computational performance of the following methods for sparse CCA estimation:

B&B: Our branch-and-bound algorithm (Algorithm 1), where the standard eigenvalue problem (13) was solved for lower- and upper-bound computations, adopting the DFS node selection and WgtUB branching rules.

Gurobi: Direct application of the optimization software Gurobi Optimizer 9.0.3 [22] to the MIO problem (14)–(18), where the indicator constraint was used to impose the constraint (16) as

$$\begin{aligned} z_j = 0 \;&\Rightarrow\; a_j = 0 \quad (j \in [p]), \\ z_{p+j} = 0 \;&\Rightarrow\; b_j = 0 \quad (j \in [q]). \end{aligned}$$

Computations were performed on a Windows computer with an Intel Core i7-4790 CPU (3.60 GHz) and 16 GB of memory, using a single thread.

L1-Rgl: L_1-regularized estimation [16], which was implemented using the `sRDA` package in the R programming language on Google Colaboratory [11]. We selected $\theta_{\boldsymbol{x}}$ and $\theta_{\boldsymbol{y}}$ variables with nonzero weights, then computed their weights by solving the corresponding generalized eigenvalue problem (28).

FwdSW: Forward stepwise selection [60], which was implemented in the Python programming language on Google Colaboratory [11].

Tables 5 and 6 show computational results of the sparse estimation methods. The columns labeled "Correlation" show values of the canonical correlation (3) for training and test sets, with best values for each problem instance are shown in bold. The column labeled "Time" shows computation times in seconds.

Table 5 lists the results for small datasets (i.e., $(p, q) \leq (13, 13)$). Our B&B algorithm attained the largest training correlation value for all problem instances. This is the expected result because our algorithm is developed for optimal sparse estimation aimed at maximizing the canonical correlation (14). On the other hand, Gurobi required much longer computation times and terminated due to the time limit for the Stdnt-M and Stdnt-P datasets. Moreover, our B&B algorithm also delivered large correlation values for the test sets. Differences in test correlation values between the B&B algorithm and other methods were especially large when subset sizes $(\theta_{\boldsymbol{x}}, \theta_{\boldsymbol{y}})$ were small. These results suggest that our algorithm is capable of not only maximizing the canonical correlation (14), but also achieving good generalization performance.

Table 6 lists the results for large datasets (i.e., $(p, q) \geq (34, 34)$). Our B&B algorithm often time-limit terminated, and optimality of the obtained solution is not guaranteed in those cases. Nevertheless, our B&B algorithm attained largest training and test correlation values for all problem instances except for the YearMSD dataset with $(\theta_{\boldsymbol{x}}, \theta_{\boldsymbol{y}}) = (10, 10)$. Differences in correlation values between the B&B algorithm and other methods were still large when subset sizes $(\theta_{\boldsymbol{x}}, \theta_{\boldsymbol{y}})$ were relatively small. These results demonstrate that our algorithm can be expected to deliver good-quality solutions even when terminated in the middle of computation.

6. Conclusion

We considered the problem of optimal sparse CCA estimation. We derived an MIO formulation for this problem and developed a high-performance branch-and-bound algorithm based on the generalized eigenvalue problem. In contrast to conventional methods for sparse CCA estimation, our algorithm is capable of finding solutions with guaranteed optimality in terms of the canonical correlation.

To confirm the effectiveness of our method, we conducted computational experiments using real-world datasets obtained from the UCI Machine Learning Repository [18]. The results suggested that our algorithm can achieve better generalization performance than do conventional methods. In addition, our algorithm can be expected to deliver good-quality solutions even when terminated in the middle of computation.

Although our method can potentially find good-quality solutions to sparse CCA problems, applying it to large datasets is computationally expensive. Thus, it is more practical to choose between our method and heuristic algorithms according to the task at hand. We also demonstrated the potential of a tailored branch-and-bound algorithm for sparse dimensionality reduction. Notably, our algorithm is a generalized version of the optimal sparse PCA algorithm [5] and can also be applied to sparse PLS regression. This will stimulate further application of optimal sparse estimation to various multivariate analyses.

A future direction of study will be to derive tighter bounds to speed up the branch-and-bound algorithm. We can also employ robust estimation techniques to mitigate the negative effects of outliers in sparse estimation algorithms. Another direction for future research is to determine optimal subset sizes $\theta_{\boldsymbol{x}}$ and $\theta_{\boldsymbol{y}}$ in the process of sparse estimation, as in the case of linear regression [41, 42, 50].

Acknowledgments

This work was partially supported by JSPS KAKENHI Grant Numbers JP21K04526 and JP21K04527.

References

[1] Akaho, S. (2001). A kernel method for canonical correlation analysis. arXiv preprint arXiv:cs/0609071v2

[2] Andrew, G., Arora, R., Bilmes, J., & Livescu, K. (2013). Deep canonical correlation analysis. In Proceedings of the 30th International Conference on Machine Learning, Proceedings of Machine Learning Research, 28(3), 1247–1255.

[3] Arthanari, T. S., & Dodge. Y. (1981). Mathematical programming in statistics. Wiley.

[4] Bach, F. R., & Jordan, M. I. (2002). Kernel independent component analysis. Journal of Machine Learning Research, 3(Jul), 1–48.

[5] Berk, L., & Bertsimas, D. (2019). Certifiably optimal sparse principal component analysis. Mathematical Programming Computation, 11(3), 381–420. https://doi.org/10.1007/s12532-018-0153-6

[6] Bertsimas, D., & King, A. (2017). Logistic regression: From art to science. Statistical Science, 32(3), 367–384. https://doi.org/10.1214/16-STS602

[7] Bertsimas, D., & King, A. (2016). OR forum—An algorithmic approach to linear regression. Operations Research, 64(1), 2–16. https://doi.org/10.1287/opre.2015.1436

[8] Bertsimas, D., King, A., & Mazumder, R. (2016). Best subset selection via a modern optimization lens. The Annals of Statistics, 813–852. https://doi.org/10.1214/15-AOS1388

[9] Bertsimas, D., & Li, M. L. (2020). Scalable holistic linear regression. Operations Research Letters, 48(3), 203–208. https://doi.org/10.1016/j.orl.2020.02.008

[10] Bertsimas, D., & Shioda, R. (2009). Algorithm for cardinality-constrained quadratic optimization. Computational Optimization and Applications, 43(1), 1–22. https://doi.org/10.1007/s10589-007-9126-9

[11] Bisong, E. (2019). Google colaboratory. In Building Machine Learning and Deep Learning Models on Google Cloud Platform (pp. 59–64). Apress. https://doi.org/10.1007/978-1-4842-4470-8_7

[12] Boyd, S., & Vandenberghe, L. (2004). Convex optimization. Cambridge University Press.

[13] Chandrashekar, G., & Sahin, F. (2014). A survey on feature selection methods. Computers & Electrical Engineering, 40(1), 16–28. https://doi.org/10.1016/j.compeleceng.2013.11.024

[14] Chen, X., Han, L., & Carbonell, J. (2012). Structured sparse canonical correlation analysis. In Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research, 22, 199–207.

[15] Cozad, A., Sahinidis, N. V., & Miller, D. C. (2014). Learning surrogate models for simulation-based optimization. AIChE Journal, 60(6), 2211–2227. https://doi.org/10.1002/aic.14418

[16] Csala, A., Voorbraak, F. P., Zwinderman, A. H., & Hof, M. H. (2017). Sparse redundancy analysis of high-dimensional genetic and genomic data. Bioinformatics, 33(20), 3228–3234. https://doi.org/10.1093/bioinformatics/btx374

[17] Dedieu, A., Hazimeh, H., & Mazumder, R. (2020). Learning sparse classifiers: Continuous and mixed integer optimization perspectives. arXiv preprint arXiv:2001.06471

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

[19] Fukumizu, K., Bach, F. R., & Gretton, A. (2007). Statistical consistency of kernel canonical correlation analysis. Journal of Machine Learning Research, 8(Feb), 361–383.

[20] Gao, C., Ma, Z., & Zhou, H. H. (2017). Sparse CCA: Adaptive estimation and computational barriers. The Annals of Statistics, 45(5), 2074–2101. https://doi.org/10.1214/16-AOS1519

[21] Gómez, A., & Prokopyev, O. (in press). A mixed-integer fractional optimization approach to best subset selection. INFORMS Journal on Computing. https://doi.org/10.1287/ijoc.2020.1031

[22] Gurobi Optimization (2020). Gurobi Optimizer Reference Manual, version 9.0, Gurobi Optimization.

[23] Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. Journal of Machine Learning Research, 3(Mar), 1157–1182.

[24] Hardoon, D. R., & Shawe-Taylor, J. (2011). Sparse canonical correlation analysis. Machine Learning, 83(3), 331–353. https://doi.org/10.1007/s10994-010-5222-7

[25] Hardoon, D. R., Szedmak, S., & Shawe-Taylor, J. (2004). Canonical correlation analysis: An overview with application to learning methods. Neural Computation, 16(12), 2639–2664. https://doi.org/10.1162/0899766042321814

[26] 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. https://doi.org/10.1214/19-STS733

[27] Hazimeh, H., Mazumder, R., & Saab, A. (2020). Sparse regression at scale: Branch-and-bound rooted in first-order optimization. arXiv preprint arXiv:2004.06152

[28] Hotelling, H. (1935). The most predictable criterion. Journal of Educational Psychology, 26(2), 139–142. https://doi.org/10.1037/h0058165

[29] Hotelling, H. (1936). Relations between two sets of variates. Biometrika, 28(3/4), 321–377. https://doi.org/10.2307/2333955

[30] Hsieh, W. W. (2000). Nonlinear canonical correlation analysis by neural networks. Neural Networks, 13(10), 1095–1105. https://doi.org/10.1016/S0893-6080(00)00067-8

[31] Kimura, K. (2019). Application of a mixed integer nonlinear programming approach to variable selection in logistic regression. Journal of the Operations Research Society of Japan, 62(1), 15–36. https://doi.org/10.15807/jorsj.62.15

[32] Kimura, K., & Waki, H. (2018). Minimization of Akaike's information criterion in linear regression analysis via mixed integer nonlinear program. Optimization Methods and Software, 33(3), 633–649. https://doi.org/10.1080/10556788.2017.1333611

[33] Konno, H., & Yamamoto, R. (2009). Choosing the best set of variables in regression analysis using integer programming. Journal of Global Optimization, 44(2), 273–282. https://doi.org/10.1007/s10898-008-9323-9

[34] Lai, P. L., & Fyfe, C. (1999). A neural implementation of canonical correlation analysis. Neural Networks, 12(10), 1391–1397. https://doi.org/10.1016/S0893-6080(99)00075-1

[35] Lê Cao, K. A., Martin, P. G., Robert-Granié, C., & Besse, P. (2009). Sparse canonical methods for biological data integration: application to a cross-platform study. BMC Bioinformatics, 10, Article 34. https://doi.org/10.1186/1471-2105-10-34

[36] Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., & Liu, H. (2017). Feature selection: A data perspective. ACM Computing Surveys, 50(6), Article 94. https://doi.org/10.1145/3136625

[37] Lin, D., Zhang, J., Li, J., Calhoun, V. D., Deng, H. W., & Wang, Y. P. (2013). Group sparse canonical correlation analysis for genomic data integration. BMC Bioinformatics, 14(1), Article 245. https://doi.org/10.1186/1471-2105-14-2

[38] Liu, H., & Motoda, H. (Eds.). (2007). Computational methods of feature selection. CRC Press.

[39] Maldonado, S., Pérez, J., Weber, R., & Labbé, M. (2014). Feature selection for support vector machines via mixed integer linear programming. Information Sciences, 279, 163–175. https://doi.org/10.1016/j.ins.2014.03.110

[40] Miller, A. (2002). Subset selection in regression. CRC Press.

[41] Miyashiro, R., & Takano, Y. (2015). Subset selection by Mallows' C_p: A mixed integer programming approach. Expert Systems with Applications, 42(1), 325–331. https://doi.org/10.1016/j.eswa.2014.07.056

[42] Miyashiro, R., & Takano, Y. (2015). Mixed integer second-order cone programming formulations for variable selection in linear regression. European Journal of Operational Research, 247(3), 721–731. https://doi.org/10.1016/j.ejor.2015.06.081

[43] Naganuma, M., Takano, Y., & Miyashiro, R. (2019). Feature subset selection for ordered logit model via tangent-plane-based approximation. IEICE Transactions on Information and Systems, 102(5), 1046–1053. https://doi.org/10.1587/transinf.2018EDP7188

[44] Park, Y. W., & Klabjan, D. (2020). Subset selection for multiple linear regression via optimization. Journal of Global Optimization, 77, 543–574. https://doi.org/10.1007/s10898-020-00876-1

[45] Parkhomenko, E., Tritchler, D., & Beyene, J. (2009). Sparse canonical correlation analysis with application to genomic data integration. Statistical Applications in Genetics and Molecular Biology, 8(1), Article 1. https://doi.org/10.2202/1544-6115.1406

[46] Silva, A. P. D. (2001). Efficient variable screening for multivariate analysis. Journal of Multivariate Analysis, 76(1), 35–62. https://doi.org/10.1006/jmva.2000.1920

[47] Saishu, H., Kudo, K., & Takano, Y. (2021). Sparse Poisson regression via mixed-integer optimization. PLOS ONE, 16(4), e0249916. https://doi.org/10.1371/journal.pone.0249916

[48] Sato, T., Takano, Y., & Miyashiro, R. (2017). Piecewise-linear approximation for feature subset selection in a sequential logit model. Journal of the Operations Research Society of Japan, 60(1), 1–14. https://doi.org/10.15807/jorsj.60.1

[49] Sato, T., Takano, Y., Miyashiro, R., & Yoshise, A. (2016). Feature subset selection for logistic regression via mixed integer optimization. Computational Optimization and Applications, 64(3), 865–880. https://doi.org/10.1007/s10589-016-9832-2

[50] Takano, Y., & Miyashiro, R. (2020). Best subset selection via cross-validation criterion. TOP, 28(2), 475–488. https://doi.org/10.1007/s11750-020-00538-1

[51] Tamura, R., Kobayashi, K., Takano, Y., Miyashiro, R., Nakata, K., & Matsui, T. (2017). Best subset selection for eliminating multicollinearity. Journal of the Operations Research Society of Japan, 60(3), 321–336. https://doi.org/10.15807/jorsj.60.321

[52] Tamura, R., Kobayashi, K., Takano, Y., Miyashiro, R., Nakata, K., & Matsui, T. (2019). Mixed integer quadratic optimization formulations for eliminating multicollinearity based on variance inflation factor. Journal of Global Optimization, 73(2), 431–446. https://doi.org/10.1007/s10898-018-0713-3

[53] Tenenhaus, A., Philippe, C., Guillemot, V., Le Cao, K. A., Grill, J., & Frouin, V. (2014). Variable selection for generalized canonical correlation analysis. Biostatistics, 15(3), 569–583. https://doi.org/10.1093/biostatistics/kxu001

[54] Thompson, B. (1984). Canonical correlation analysis: Uses and interpretation. SAGE Publications.

[55] Tuzhilina, E., Tozzi, L., & Hastie, T. (2020). Canonical correlation analysis in high dimensions with structured regularization. arXiv preprint arXiv:2011.01650

[56] Uurtio, V., Monteiro, J. M., Kandola, J., Shawe-Taylor, J., Fernandez-Reyes, D., & Rousu, J. (2017). A tutorial on canonical correlation methods. ACM Computing Surveys, 50(6), Article 95. https://doi.org/10.1145/3136624

[57] Ustun, B., & Rudin, C. (2016). Supersparse linear integer models for optimized medical scoring systems. Machine Learning, 102(3), 349–391. https://doi.org/10.1007/s10994-015-5528-6

[58] Waaijenborg, S., de Witt Hamer, P. V., & Zwinderman, A. H. (2008). Quantifying the association between gene expressions and DNA-markers by penalized canonical correlation analysis. Statistical Applications in Genetics and Molecular Biology, 7(1), Article 3. https://doi.org/10.2202/1544-6115.1329

[59] Wang, W., Arora, R., Livescu, K., & Bilmes, J. (2015). On deep multi-view representation learning. In Proceedings of the 32nd International Conference on Machine Learning, Proceedings of Machine Learning Research, 37, 1083–1092.

[60] Wiesel, A., Kliger, M., & Hero III, A. O. (2008). A greedy approach to sparse canonical correlation analysis. arXiv preprint arXiv:0801.2748

[61] Williams, H. P. (2013). Model building in mathematical programming. John Wiley & Sons.

[62] Wilms, I., & Croux, C. (2015). Sparse canonical correlation analysis from a predictive point of view. Biometrical Journal, 57(5), 834–851. https://doi.org/10.1002/bimj.201400226

[63] Witten, D. M., & Tibshirani, R. J. (2009). Extensions of sparse canonical correlation analysis with applications to genomic data. Statistical Applications in Genetics and Molecular Biology, 8(1), Article 28. https://doi.org/10.2202/1544-6115.1470

[64] Witten, D. M., Tibshirani, R., & Hastie, T. (2009). A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. Biostatistics, 10(3), 515–534. https://doi.org/10.1093/biostatistics/kxp008

[65] Wolsey, L. A., & Nemhauser, G. L. (1999). Integer and Combinatorial Optimization. John Wiley & Sons.

[66] Yan, L., & Chen, X. (2020). Certifiably optimal sparse sufficient dimension reduction. arXiv preprint arXiv:2012.08065

[67] Zhuang, X., Yang, Z., & Cordes, D. (2020). A technical review of canonical correlation analysis for neuroscience applications. Human Brain Mapping, 41(13), 3807–3833. https://doi.org/10.1002/hbm.25090

Algorithm 1 Branch-and-bound Algorithm for Solving Problem (14)–(18).

Input: covariance matrices $\boldsymbol{C_{xx}}, \boldsymbol{C_{xy}}, \boldsymbol{C_{yy}}$; subset sizes $\theta_{\boldsymbol{x}}, \theta_{\boldsymbol{y}}$; optimality tolerance ε

Output: ε-optimal solution $(\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{z}})$ to problem (14)–(18)

initialization: $(\boldsymbol{\ell}, \boldsymbol{u}) \leftarrow (\mathbf{0}, \mathbf{1})$, $\mathcal{N} \leftarrow \{(\boldsymbol{\ell}, \boldsymbol{u})\}$, $(\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{z}}) \leftarrow (\mathbf{0}, \mathbf{0}, \mathbf{0})$, $f^{\mathrm{LB}} \leftarrow 0$, $f^{\mathrm{UB}} \leftarrow \lambda^*_{\max}$
while $f^{\mathrm{UB}} - f^{\mathrm{LB}} > \varepsilon$ **do** ▷ optimality gap
  set $k \leftarrow 0$, select $(\boldsymbol{\ell}, \boldsymbol{u}) \in \mathcal{N}$, and remove $(\boldsymbol{\ell}, \boldsymbol{u})$ from $\mathcal{N}$ ▷ by a node selection rule
  if $terminal((\boldsymbol{\ell}, \boldsymbol{u}), (1, p), \theta_{\boldsymbol{x}}) = false$ **then** ▷ Eq. (26)
    select $k \in [p]$ such that $(\ell_k, u_k) = (0, 1)$ ▷ by a branching rule
  else if $terminal((\boldsymbol{\ell}, \boldsymbol{u}), (p+1, p+q), \theta_{\boldsymbol{y}}) = false$ **then** ▷ Eq. (27)
    select $k - p \in [q]$ such that $(\ell_k, u_k) = (0, 1)$ ▷ by a branching rule
  if $k = 0$ **then** ▷ $(\boldsymbol{l}, \boldsymbol{u})$ is a terminal node
    compute $lower \leftarrow \lambda^*_{\max}(\boldsymbol{z}^{\mathrm{SP}})$ with $(\boldsymbol{a}^{\mathrm{SP}}, \boldsymbol{b}^{\mathrm{SP}}, \boldsymbol{z}^{\mathrm{SP}})$ ▷ Eqs. (29)–(31)
    if $lower > f^{\mathrm{LB}}$ **then**
      update $f^{\mathrm{LB}} \leftarrow lower$ and $(\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{z}}) \leftarrow (\boldsymbol{a}^{\mathrm{SP}}, \boldsymbol{b}^{\mathrm{SP}}, \boldsymbol{z}^{\mathrm{SP}})$ ▷ lower-bound update
  else ▷ $(\boldsymbol{l}, \boldsymbol{u})$ is not a terminal node
    for $val \in \{0, 1\}$ **do**
      set $newnode \leftarrow (\boldsymbol{\ell}, \boldsymbol{u})$ with $\ell_k = u_k = val$ ▷ branching operation
      if $val = 0$ **then** ▷ $z_k = 0$
        compute $upper \leftarrow upper(newnode)$ ▷ Eq. (32)
      else if $val = 1$ **then** ▷ $z_k = 1$
        take $upper$ from $newnode$'s parent node
      compute $lower \leftarrow lower(newnode)$ with $(\boldsymbol{a}^{\mathrm{LB}}, \boldsymbol{b}^{\mathrm{LB}}, \boldsymbol{z}^{\mathrm{LB}})$ ▷ Eq. (33)
      if $lower > f^{\mathrm{LB}}$ **then**
        update $f^{\mathrm{LB}} \leftarrow lower$ and $(\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{z}}) \leftarrow (\boldsymbol{a}^{\mathrm{LB}}, \boldsymbol{b}^{\mathrm{LB}}, \boldsymbol{z}^{\mathrm{LB}})$ ▷ lower-bound update
      if $upper > f^{\mathrm{LB}}$ **then**
        add $newnode$ to $\mathcal{N}$
  remove any $node$ from $\mathcal{N}$ such that $upper(node) \leq f^{\mathrm{LB}}$ ▷ bounding operation
  update $f^{\mathrm{UB}} \leftarrow \max\{upper(node) \mid node \in \mathcal{N}\}$ ▷ upper-bound update
return $(\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{z}})$

Table 2: Comparison of Eigenvalue Problems

Dataset	n	p	q	$\theta_{\boldsymbol{x}}$	$\theta_{\boldsymbol{y}}$	Methods	LB	Gap	#Nodes	Time
Stdnt-M	395	13	13	3	3	GenEgv	0.4735	**0%**	4,971	3.16
						StdEgv	0.4735	**0%**	4,971	3.06
				5	5	GenEgv	0.5401	**0%**	6,379	4.76
						StdEgv	0.5401	**0%**	6,379	4.31
Stdnt-P	649	13	13	3	3	GenEgv	0.4619	**0%**	2,727	1.74
						StdEgv	0.4619	**0%**	2,727	1.60
				5	5	GenEgv	0.5081	**0%**	3,313	2.47
						StdEgv	0.5081	**0%**	3,313	2.22
Music	1,059	34	34	3	3	GenEgv	0.8693	**0%**	71,083	68.78
						StdEgv	0.8693	**0%**	71,083	56.54
				5	5	GenEgv	0.8799	5.6%	722,919	$> 1{,}000$
						StdEgv	0.8808	**5.5%**	1,043,617	$> 1{,}000$
				10	10	GenEgv	0.9094	**2.1%**	612,003	$> 1{,}000$
						StdEgv	0.9094	**2.1%**	997,873	$> 1{,}000$
YearMSD	515,345	45	45	3	3	GenEgv	0.7755	**0%**	371,261	475.52
						StdEgv	0.7755	**0%**	371,261	403.62
				5	5	GenEgv	0.8052	**11.9%**	602,495	$> 1{,}000$
						StdEgv	0.8052	**11.9%**	784,345	$> 1{,}000$
				10	10	GenEgv	0.8495	**6.1%**	535,823	$> 1{,}000$
						StdEgv	0.8495	**6.1%**	755,025	$> 1{,}000$

Table 3: Comparison of Node Selection and Branching Rules ($(p, q) \leq (13, 13)$)

Dataset	n	p	q	$\theta_{\boldsymbol{x}}$	$\theta_{\boldsymbol{y}}$	Node	Branch	LB	Gap	#Nodes	Time
Stdnt-M	395	13	13	3	3	BFS	Random	0.4735	**0%**	47,089	565.1
							WgtFull	0.4735	**0%**	7,909	19.3
							WgtUB	0.4735	**0%**	3,501	4.8
						DFS	Random	0.4735	**0%**	58,265	34.4
							WgtFull	0.4735	**0%**	8,415	4.9
							WgtUB	0.4735	**0%**	4,971	3.1
						LUB	Random	0.4735	**0%**	44,339	518.8
							WgtFull	0.4735	**0%**	7,909	17.2
							WgtUB	0.4735	**0%**	3,501	5.9
				5	5	BFS	Random	0.5367	9.8%	37,045	$> 1{,}000$
							WgtFull	0.5401	**0%**	5,779	10.5
							WgtUB	0.5401	**0%**	3,493	5.4
						DFS	Random	0.5401	**0%**	135,527	92.5
							WgtFull	0.5401	**0%**	5,919	3.7
							WgtUB	0.5401	**0%**	6,379	4.3
						LUB	Random	0.5401	1.9%	41,567	$> 1{,}000$
							WgtFull	0.5401	**0%**	5,511	10.1
							WgtUB	0.5401	**0%**	3,193	4.9
Stdnt-P	649	13	13	3	3	BFS	Random	0.4619	**0%**	33,859	303.7
							WgtFull	0.4619	**0%**	6,697	14.8
							WgtUB	0.4619	**0%**	2,501	3.2
						DFS	Random	0.4619	**0%**	38,627	23.9
							WgtFull	0.4619	**0%**	6,743	3.8
							WgtUB	0.4619	**0%**	2,727	1.6
						LUB	Random	0.4619	**0%**	39,705	358.7
							WgtFull	0.4619	**0%**	6,697	11.8
							WgtUB	0.4619	**0%**	2,501	3.1
				5	5	BFS	Random	0.5075	4.5%	43,139	$> 1{,}000$
							WgtFull	0.5081	**0%**	3,163	4.4
							WgtUB	0.5081	**0%**	1,785	2.0
						DFS	Random	0.5081	**0%**	117,123	81.6
							WgtFull	0.5081	**0%**	3,113	2.0
							WgtUB	0.5081	**0%**	3,313	2.2
						LUB	Random	0.5081	0.9%	42,737	$> 1{,}000$
							WgtFull	0.5081	**0%**	3,075	3.8
							WgtUB	0.5081	**0%**	1,701	1.9

Table 4: Comparison of Node Selection and Branching Rules ($(p, q) \geq (34, 34)$)

Dataset	n	p	q	$\theta_{\boldsymbol{x}}$	$\theta_{\boldsymbol{y}}$	Node	Branch	LB	Gap	#Nodes	Time
Music	1,059	34	34	3	3	BFS	Random	0.8660	7.2%	25,649	> 1,000
							WgtFull	0.8660	5.8%	26,009	> 1,000
							WgtUB	0.8693	4.3%	44,183	> 1,000
						DFS	Random	0.8647	7.4%	821,135	> 1,000
							WgtFull	0.8618	5.7%	1,287,523	> 1,000
							WgtUB	0.8692	**0%**	71,083	56.5
						LUB	Random	0.8660	6.6%	32,903	> 1,000
							WgtFull	0.8681	3.3%	34,135	> 1,000
							WgtUB	0.8692	0.4%	44,909	> 1,000
				5	5	BFS	Random	0.8869	4.7%	20,291	> 1,000
							WgtFull	0.8869	4.3%	20,537	> 1,000
							WgtUB	0.8908	3.9%	23,203	> 1,000
						DFS	Random	0.8804	5.5%	794,145	> 1,000
							WgtFull	0.8748	5.8%	1,192,177	> 1,000
							WgtUB	0.8808	5.5%	1,043,617	> 1,000
						LUB	Random	0.8805	5.3%	24,235	> 1,000
							WgtFull	0.8837	3.3%	41,771	> 1,000
							WgtUB	0.8869	**2.6%**	40,213	> 1,000
YearMSD	515,345	45	45	3	3	BFS	Random	0.7753	16.1%	22,755	> 1,000
							WgtFull	0.7755	11.7%	22,777	> 1,000
							WgtUB	0.7755	11.4%	23,055	> 1,000
						DFS	Random	0.7753	16.3%	657,839	> 1,000
							WgtFull	0.7755	14.4%	1,062,445	> 1,000
							WgtUB	0.7755	**0%**	371,261	403.6
						LUB	Random	0.7753	15.5%	26,449	> 1,000
							WgtFull	0.7755	7.4%	30,107	> 1,000
							WgtUB	0.7755	5.6%	30,927	> 1,000
				5	5	BFS	Random	0.8160	10.5%	18,251	> 1,000
							WgtFull	0.8172	9.4%	18,433	> 1,000
							WgtUB	0.8172	9.3%	18,857	> 1,000
						DFS	Random	0.8052	11.9%	631,163	> 1,000
							WgtFull	0.8172	9.8%	970,483	> 1,000
							WgtUB	0.8052	11.9%	784,345	> 1,000
						LUB	Random	0.8052	11.6%	21,183	> 1,000
							WgtFull	0.8172	6.2%	24,649	> 1,000
							WgtUB	0.8172	**5.7%**	24,335	> 1,000

Table 5: Comparison of Sparse Estimation Methods ($(p, q) \leq (13, 13)$)

							Correlation		
Dataset	n	p	q	$\theta_{\boldsymbol{x}}$	$\theta_{\boldsymbol{y}}$	Method	Training	Test	Time
Wine-R	1,599	6	5	2	2	B&B	**0.8477**	**0.8529**	< 0.1
						Gurobi	**0.8477**	**0.8529**	2.1
						L1-Rgl	0.6808	0.6103	0.1
						FwdSW	0.8208	0.8288	< 0.1
				3	3	B&B	**0.9304**	**0.8545**	< 0.1
						Gurobi	**0.9304**	**0.8545**	46.9
						L1-Rgl	0.8525	0.8460	0.2
						FwdSW	0.9247	0.8456	< 0.1
Wine-W	4,898	6	5	2	2	B&B	**0.9248**	**0.8372**	< 0.1
						Gurobi	**0.9248**	**0.8372**	1.5
						L1-Rgl	0.8668	0.7846	0.2
						FwdSW	0.9033	0.8201	< 0.1
				3	3	B&B	**0.9601**	**0.8751**	0.1
						Gurobi	**0.9601**	**0.8751**	37.3
						L1-Rgl	0.9254	0.8340	0.3
						FwdSW	**0.9601**	**0.8751**	< 0.1
Stdnt-M	395	13	13	3	3	B&B	**0.4735**	0.4984	3.1
						Gurobi	0.4264	**0.5674**	$> 1{,}000$
						L1-Rgl	0.4042	0.1054	< 0.1
						FwdSW	0.4443	0.4443	< 0.1
				5	5	B&B	**0.5401**	0.4932	4.3
						Gurobi	0.4885	**0.4952**	$> 1{,}000$
						L1-Rgl	0.4920	0.3633	0.1
						FwdSW	**0.5401**	0.4932	< 0.1
Stdnt-P	649	13	13	3	3	B&B	**0.4619**	**0.3466**	1.6
						Gurobi	0.4502	0.3183	$> 1{,}000$
						L1-Rgl	0.4585	0.3141	< 0.1
						FwdSW	0.4595	0.3148	< 0.1
				5	5	B&B	**0.5081**	0.3324	2.2
						Gurobi	0.4800	0.3073	$> 1{,}000$
						L1-Rgl	0.5042	**0.3394**	0.3
						FwdSW	**0.5081**	0.3324	< 0.1

Table 6: Comparison of Sparse Estimation Methods ($(p, q) \geq (34, 34)$)

							Correlation		
Dataset	n	p	q	$\theta_{\boldsymbol{x}}$	$\theta_{\boldsymbol{y}}$	Method	Training	Test	Time
Music	1,059	34	34	3	3	B&B	**0.8693**	**0.6335**	56.5
						Gurobi	0.6788	0.6195	> 1,000
						L1-Rgl	0.7999	0.6130	0.2
						FwdSW	0.8258	0.5188	0.1
				5	5	B&B	**0.8808**	**0.7043**	> 1,000
						Gurobi	0.6572	0.5502	> 1,000
						L1-Rgl	0.8122	0.5633	0.2
						FwdSW	0.8501	0.5617	0.2
				10	10	B&B	**0.9094**	**0.9379**	> 1,000
						Gurobi	N/A	N/A	> 1,000
						L1-Rgl	0.8274	0.6200	0.2
						FwdSW	0.8895	0.6570	0.6
YearMSD	515,345	45	45	3	3	B&B	**0.7755**	**0.8027**	403.6
						Gurobi	0.4877	0.5064	> 1,000
						L1-Rgl	0.6402	0.6395	130.9
						FwdSW	0.7280	0.7366	9.6
				5	5	B&B	**0.8052**	**0.8303**	> 1,000
						Gurobi	0.5575	0.5693	> 1,000
						L1-Rgl	0.6971	0.6982	133.9
						FwdSW	0.7910	0.8024	11.0
				10	10	B&B	0.8495	0.8650	> 1,000
						Gurobi	0.6530	0.6657	> 1,000
						L1-Rgl	0.7642	0.7566	193.9
						FwdSW	**0.8663**	**0.8873**	115.2