Principal Component Hierarchy for Sparse Quadratic Programs

by

Robbie Vreugdenhil

to obtain the degree of Master of Science in Systems and Control
at the Delft University of Technology,
to be defended publicly on Tuesday August 24, 2021 at 09:30 AM.

Student number:	4438035	
Project duration:	September 1, 2020 – August 10, 2021	
Thesis committee:	Dr. P. Mohajerin Esfahani,	TU Delft, supervisor
	Dr. Ing. J. Kober,	TU Delft
	Dr. B. Hunyadi,	TU Delft

TU Delft

Preface

This documents consists of two separate papers. The first chapter covers 'Principal component hierarchy for sparse quadratic programming problems', which has been published at the International Conference of Machine Learning 2021. The second chapter covers 'Sparse inductive matrix completion using updated side information'.
I would like to thank Viet Anh Nguyen and Armin Eftekhari for their contributions to the work in chapter 1. Above all I want to thank my supervisor Peyman Mohajerin Esfahani for his insights and the great discussions we had.

Robbie Vreugdenhil
Delft, August 2021

Contents

Chapter 1

Principal Component Hierarchy for Sparse Quadratic Programs

Abstract

We propose a novel approximation hierarchy for cardinality-constrained, convex quadratic programs that exploits the rank-dominating eigenvectors of the quadratic matrix. Each level of approximation admits a min-max characterization whose objective function can be optimized over the binary variables analytically, while preserving convexity in the continuous variables. Exploiting this property, we propose two scalable optimization algorithms, coined as the "*best response*" and the "*dual program*", that can efficiently screen the potential indices of the nonzero elements of the original program. We show that the proposed methods are competitive with the existing screening methods in the current sparse regression literature, and it is particularly fast on instances with high number of measurements in experiments with both synthetic and real datasets.

1 Introduction

Sparsity is a powerful inductive bias that improves the interpretability and performance of many regression models [Ribeiro et al., 2016, Hastie et al., 2009, Bertsimas et al., 2017]. Recent years have witnessed a growing interest in sparsity-based methods and algorithms for sparse recovery, mostly in the setting of sparse linear regression [Atamturk and Gomez, 2020, Bertsimas and van Parys, 2017, Hazimeh et al., 2020, Hastie et al., 2017].

In this paper we study the more general problem of sparse linearly-constrained quadratic programming with a regularization term. Sparsity is imposed in this context by controlling the ℓ_0 norm of

the estimator [Miller, 2002]. More specifically, we consider the problem

$$\begin{array}{rll} J^\star \triangleq & \min & \langle c, x\rangle + \langle x, Qx\rangle + \eta^{-1}\|x\|_2^2 \\ & \text{s.t.} & x \in \mathbb{R}^n \\ & & Ax \leq b, \quad \|x\|_0 \leq s, \end{array} \tag{$\mathcal{P}$}$$

where $Q \in \mathbb{S}_+^n$, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, and the integer $s \leq n$ specifies the target sparsity level. The ridge regularization term $\eta^{-1}\|x\|_2^2$ [Hoerl and Kennard, 1970] in the objective function reduces the mean squared error when the data is affected by noise and/or uncertainty [Ghaoui and Lebret, 1997, Mazumder et al., 2020].

Problem ($\mathcal{P}$) arises in a wide range of applications, including sparse linear regression, model predictive control [Aguilera et al., 2014], portfolio optimization [Bertsimas and Cory-Wright, 2020], binary quadratic programming and principal component analysis [Bertsimas et al., 2020b].

Motivation for our approach. Throughout, suppose that Q in problem ($\mathcal{P}$) admits the eigendecomposition $Q = \sum_{i=1}^n \lambda_i v_i v_i^\top$, where $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$ are the eigenvalues of Q. Our approach to solve ($\mathcal{P}$) is centered around the following key observation:

> **Observation.** For many real-world applications, the matrix Q is *nearly low-rank*.

The concept of *nearly low-rank* is contextual. In this paper, we say that Q is nearly low-rank if a low-rank matrix can approximate Q to a reasonable accuracy, where the accuracy is measured by a matrix norm.

It is instructive to demonstrate the nearly low-rank property of Q in the context of linear regression. Given N training samples $\{(\xi_i, \omega_i)\}_{i=1}^N \subset \mathbb{R}^n \times \mathbb{R}$, the sparse ridge regression problem

$$\min_{x:\|x\|_0\leq s} \quad \frac{1}{N}\sum_{i=1}^N \|\xi_i - x^\top \omega_i\|_2^2 + \eta^{-1}\|x\|_2^2 \tag{1.1}$$

coincides with ($\mathcal{P}$) for the choice of

$$Q = \frac{1}{N}\sum_{i=1}^N \omega_i\omega_i^\top, \quad c = \frac{1}{N}\sum_{i=1}^N \xi_i\omega_i. \tag{1.2}$$

In high-dimensional regression ($N < n$), the matrix Q is automatically rank-deficient. Moreover, Q can also be nearly low-rank even when $N > n$. As a numerical example, for the UCI Superconductivity dataset[1], we randomly select 70% of the dataset as training samples, and then compute the matrix Q, as specified in (1.2). Figure 1.1 plots the empirical distribution of the largest eigenvalues of Q, taken over 100 independent replications. On average, the ratio λ_1/λ_{10} between the 1$^{\text{st}}$

[1]Available at https://archive.ics.uci.edu/ml/datasets/Superconductivty+Data

largest and the 10^{th} largest eigenvalues of Q is 97.2. We can also observe that the magnitude of the eigenvalues decays relatively fast for this dataset.

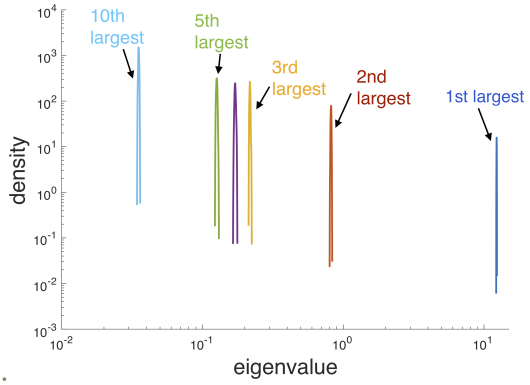


Figure 1.1: Empirical distribution of the eigenvalues of the matrix Q for the Superconductivity dataset (plotted in log-log scale).

Next, let us recall the low-rank truncation of the matrix Q. For $k \leq n$, the approximation of Q using its k leading eigenvectors, denoted by Q_k, is given by

$$Q_k \triangleq \sum_{i=1}^{k} \lambda_i v_i v_i^\top.$$

Following the same sampling procedure described above, Figure 1.2 depicts how the average truncation error, measured by the Frobenius norm $\|Q_k - Q\|_F$, rapidly decreases as k increases. In this figure, we also report the minimum dimension $\hat{k}$ so that the reconstruction error of $Q_{\hat{k}}$ falls below 10% that of the rank-1 approximation Q_1. We observe that $\hat{k} \ll n$ for many UCI regression datasets. This nearly low-rank behavior is typical in data sciences [Udell and Townsend, 2019].

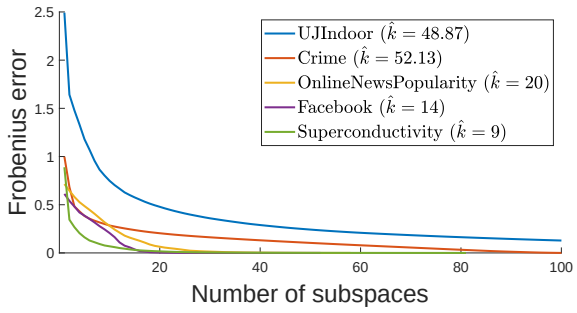


Figure 1.2: Empirical Frobenius reconstruction error $\|Q_k - Q\|_F$ averaged over 100 independent replications.

Contributions. We summarize the contributions as follows.

(i) **Hierarchy of approximations and min-max characterization:** We exploit the nearly low-rank nature of the matrix Q and propose a hierarchy of approximations for cardinality-constrained convex quadratic programs. This hierarchy enables us to strike a balance between the scalability of the solver and the quality of the solution. We further show that each k-leading spectral truncation can be characterized as a min-max problem (Proposition 3.1).

(ii) **Scalable and certifiable algorithms:** We propose two scalable algorithms that enjoy optimality certificates if the min-max characterization admits a saddle point (Propositions 3.3 and 3.5). The proposed algorithms build on a desirable feature of the objective function of the min-max characterization, in which the minimizer over the binary variables admits a closed-form solution (Lemma 3.2).

(iii) **Safe screening:** If the min-max characterizations do not admit a saddle point, the proposed iterative algorithms can serve as a screening method to reduce the variables in the original problem. We investigate the effectiveness of our algorithms through an in-depth numerical comparison with the recent sparse regression literature including the safe screening procedures of Atamturk and Gomez [2020] and the warm start of Bertsimas and van Parys [2017]. Moreover, we also benchmark against direct optimization methods Beck and Eldar [2012] and Yuan et al. [2020]. Experiments on both synthetic and real datasets reveal that our algorithms deliver promising results (Section 4).

We note that using leading principal components in regression has a long-standing history [Næs and Martens, 1988, Hastie et al., 2009, Baye and Parker, 1984]. However, to the best of our knowledge, it has never been applied in the context of cardinality-constrained convex quadratic problems. Compared to the existing solution procedures in the literature, the performance of our methods, measured by both the objective value and the screening capacity, is consistent across a wider range of input parameters. Moreover, in sparse regression, our methods can scale to problems of large sample size N because the complexity of our methods does not depend on N poorly.

This paper unfolds as follows. Section 2 provides a brief overview over the landscape of the cardinality-constrained quadratic problems. Section 3 devises two distinctive algorithms that leverage the principal component approximation of the matrix Q. Section 4 reports an in-depth numerical comparison between our algorithms and the current sparse regression literature.

Notations. For any matrix u, we use $\sqrt{u}$ and $|u|$ to denote the component-wise square root and absolute value of u, respectively. For a vector u, we use $\mathrm{diag}(u)$ to denote the diagonal matrix formed by u. For an integer n, we also denote with $[n]$ the set of integers $\{1, \ldots, n\}$. Given an

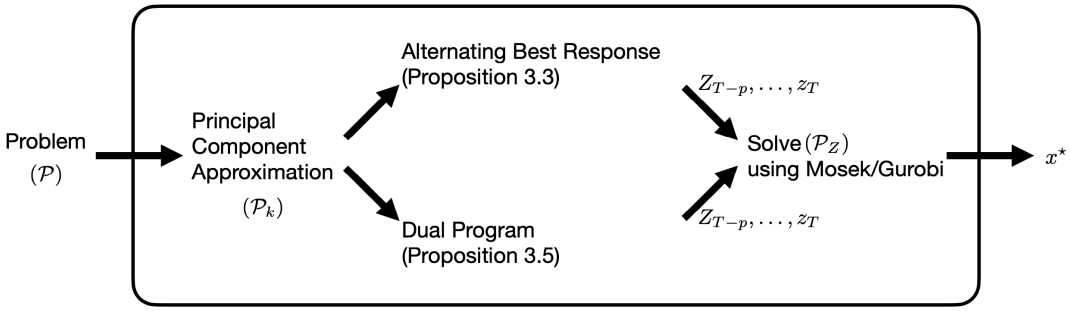


Figure 1.3: Schematic overview of the principal component approximation to sparsity-constrained quadratic programs.

index set $\mathbb{A} \subset [n]$, the binary vector $u = \mathbb{1}_n(\mathbb{A}) \in \{0,1\}^n$ is defined as $u_j = 1$ if and only if $j \in \mathbb{A}$. In words, $\mathbb{1}_n(\mathbb{A})$ is the indicator vector of the index set $\mathbb{A}$.

2 Literature Review

Sparse regression. In general, the sparse regression problem is NP-hard [Natarajan, 1995]. Until recently the sparse optimization literature largely focused on convex heuristics for sparse regression, e.g., Lasso (ℓ_1) [Tibshirani, 1996] and Elastic Net ($\ell_1 - \ell_2$) [Zou and Hastie, 2005]. Despite their scalability, convex heuristic approaches are inherently biased since the ℓ_1-norm penalizes large and small coefficients uniformly. In contrast, solvers that directly tackle sparse regression do not suffer from unwanted shrinkage and have enjoyed a resurgence of interest [Bertsimas and van Parys, 2017, Hazimeh et al., 2020, Dedieu et al., 2020, Gomez and Prokopyev, 2018], thanks to the recent breakthroughs in mixed-integer programming [Bertsimas et al., 2016]. These direct approaches are also the focus of this paper.

In particular, Bertsimas and van Parys [2017] devise a convex reformulation of the sparse regression problem using duality theory, the solution of which provides a warm start for a branch-and-cut algorithm. This method can solve sparse regression problems at the scale of $n \approx 10^5$ while the earlier work [Bertsimas et al., 2016] only goes to sizes of $n \approx 10^3$. However, as pointed out by Xie and Deng [2020], the performance of these algorithms depends critically on the speed of the commercial solvers and varies significantly from one dataset to another. Therefore we focus on the warm start method in the approach of Bertsimas and van Parys [2017], which makes use of the kernel matrix $[\omega_i^\top \omega_j]_{i,j}$. Notice that the size of this kernel matrix scales with the number of samples. On the contrary, our proposed solution procedures use only the resulting matrix Q, whose size does not depend on the number of samples.

Convex approximations of the sparse regression can however be used as a safe screening procedure as demonstrated by Atamturk and Gomez [2020]. Safe screening methods aim to identify the support of the solution set of ($\mathcal{P}$) and reduce the problem dimension n before invoking an MIQP solver. Indeed, if we correctly rule out any single suboptimal dimension, the solution space would be cut by half. In this sense, the expected speedup for the MIQP solver is exponential [Atamturk and Gomez, 2020].

Sparse quadratic programs. We are only aware of a few papers that solve sparse quadratic programs exactly. Beck and Eldar [2012] and Beck and Hallak [2015] devise coordinate descent type algorithms based on the concept of coordinate-wise optimality, which updates the support at each iteration and, in particular, can also be used to solve sparse quadratic programs. Yuan et al. [2020] considers a block decomposition algorithm that combines combinatorial search and coordinate descent. Specifically, this method uses a random or greedy strategy to find the working set and then performs a global combinatorial search over the working set based on the original objective function. Recently, Bertsimas and Cory-Wright [2020] and Bertsimas et al. [2020b] apply the advances in exact sparse regression to sparse quadratic programs to solve problems of higher dimension. Both essentially rewrite ($\mathcal{P}$) as a sparse regression problem and then solve it with a modified version of the branch-and-cut algorithm of Bertsimas and van Parys [2017]. This procedure can also accommodate linear constraints.

Given the existing literature mentioned above, to our best of knowledge, our approach is the first to identify and exploit the low-rank structure of Q in the original problem ($\mathcal{P}$) by using the leading principal component approximation of Q, notably in the context of sparse quadratic programming.

3 Principal Component Approximation

The key idea of this study is to leverage the principal component approximation of the matrix Q in ($\mathcal{P}$) in order to deploy the duality technique from convex optimization in a more efficient manner. To this end, we introduce additional continuous variables y along with the equality constraints $\sqrt{\lambda_i} y_i = \sqrt{\lambda_i} \langle v_i, x \rangle$ where λ_i and v_i are, respectively, the eigenvalues and eigenvectors of the matrix Q. Throughout, we denote $V = [v_1, \ldots, v_k] \in \mathbb{R}^{d \times k}$. Using these definitions, program ($\mathcal{P}$)

can be approximated via

$$
\begin{aligned}
J_k^\star \triangleq \quad & \min \quad \langle c, x\rangle + \sum_{i=1}^{k} \lambda_i y_i^2 + \eta^{-1}\|x\|_2^2 \\
& \text{s.t.} \quad x \in \mathbb{R}^n,\ y \in \mathbb{R}^k \\
& \qquad Ax \le b, \quad \|x\|_0 \le s \\
& \qquad \sqrt{\lambda_i} y_i = \sqrt{\lambda_i}\,\langle v_i, x\rangle\,, \quad i \in [k].
\end{aligned}
\tag{$\mathcal{P}_k$}
$$

The last equality constraints of ($\mathcal{P}_k$) are scaled using the coefficients $\sqrt{\lambda_i}$ to improve numerical stability. Since the matrix Q is positive semidefinite, we have the hierarchy of approximations in which the sequence of optimal values $J_k^\star$ preserves the order

$$J_1^\star \le J_2^\star \le \cdots \le J_n^\star = J^\star,$$

where $J^\star$ is the optimal value of program ($\mathcal{P}$). In fact, one can observe that the two programs ($\mathcal{P}$) and ($\mathcal{P}_k$) are equivalent when $k = n$. Next, we use the standard convex duality to turn program ($\mathcal{P}_k$) into a min-max optimization problem.

Proposition 3.1 (Min-max characterization)**.** *For each $k \le n$, the optimal value $J_k^\star$ of ($\mathcal{P}_k$) is equal to*

$$J_k^\star = \min_{\substack{z\in\{0,1\}^n \\ \sum z_j \le s}} \ \max_{\substack{\alpha\in\mathbb{R}^k \\ \beta\in\mathbb{R}^m_+}} \ L(z,\alpha,\beta), \tag{$\mathcal{M}_k$}$$

where the objective function L is defined as

$$L(z,\alpha,\beta) \triangleq -\beta^\top b - \frac{1}{4}\|\alpha\|_2^2 - \frac{\eta}{4}(c + V\sqrt{\Lambda}\alpha + A^\top\beta)^\top \mathrm{diag}(z)(c + V\sqrt{\Lambda}\alpha + A^\top\beta), \tag{1.3}$$

in which $\Lambda = \mathrm{diag}\{\lambda_1, \cdots, \lambda_k\}$ is a diagonal matrix whose elements on the main diagonal are the first k largest eigenvalues of the matrix Q. Moreover, the nonzero coordinates of the optimal variable $x^\star$ in ($\mathcal{P}_k$) contain the nonzero elements of the optimal solution $z^\star$ in ($\mathcal{M}_k$).

Thanks to Proposition 3.1, the information regarding the support of the optimal solution $x^\star$ in ($\mathcal{P}_k$) can effectively be obtained from the support of the optimal solution $z^\star$ in ($\mathcal{M}_k$). We note that finding the support is indeed the computational bottleneck of program ($\mathcal{P}_k$). A key feature of the objective function L of program ($\mathcal{M}_k$) is that when the variables (α, β) is fixed, the minimization over the binary variable z can be solved analytically.

Lemma 3.2 (Closed-form minimizer)**.** *Given any pair (α, β), the minimizer of the function L defined in (1.3) can be computed as*

$$\arg\min_{\substack{z\in\{0,1\}^n \\ \sum z_j \le s}} L(z,\alpha,\beta) = \mathbb{1}_n\big(\mathcal{J}(\alpha,\beta)\big), \quad \textit{where} \tag{1.4}$$

$$\mathcal{J}(\alpha,\beta) \triangleq \left\{ j \in [n] : \begin{array}{c} j \text{ is an index of the } s\text{-largest} \\ \text{elements of the vector} \\ c + V\sqrt{\Lambda}\alpha + A^\top\beta \end{array} \right\},$$

and $\mathbb{1}_n(\mathcal{J}) \in \{0,1\}^n$ *is a binary vector whose coordinates contained in the set* $\mathcal{J}$ *are ones.*

We note that the set of optimal indices defined in $\mathcal{J}(\alpha,\beta)$ may not be unique. In such scenarios, we adopt a deterministic tiebreaker (e.g., a lexicographic rule) to introduce $\mathcal{J}$ as a proper single-valued function. The observation in (1.4) is the key building block for two scalable optimization algorithms that we will propose to tackle problem ($\mathcal{P}_k$).

3.1 Alternating best response

The first proposed algorithm can be cast as an attempt to find a saddle point (a Nash equilibrium) of program ($\mathcal{M}_k$), if it exists. This is similar to the approach discussed in Bertsimas et al. [2020a, Theorem 3.2.3], which concerns the different problem of sparse PCA. Given the inherent nonconvexity of the problem due to the binary variables z, such an equilibrium may not exist. In that case, the algorithm will not converge. Nonetheless, we will also discuss how this approach can still be viewed as a "safe screening" scheme [Atamturk and Gomez, 2020].

Given $z \in \{0,1\}^n$, we define the function

$$\mathrm{BR}(z) \triangleq \arg\max_{\substack{\alpha\in\mathbb{R}^k \\ \beta\in\mathbb{R}^m_+}} L(z,\alpha,\beta). \tag{1.5}$$

If the function L in (1.5) does not have a unique optimizer, in a similar fashion as as $\mathcal{J}$ defined in Lemma 3.2, we deploy a deterministic tiebreaker to properly introduce a single-valued function BR. The function BR is the maximizer of the loss function for a fixed value of z, to which we refer as the "best response". While the objective function L is a jointly concave quadratic function in the variables (α,β), the description of the optimizer does not necessarily have an explicit description due to the constraint $\beta \geq 0$. However, in the absence of the constraint $Ax \leq b$ in ($\mathcal{P}_k$) (e.g., $A = 0, b = 0$), the function (1.5) can be explicitly described as

$$\mathrm{BR}(z) = -(I_k/\eta + \sqrt{\Lambda}V^\top \mathrm{diag}(z) V\sqrt{\Lambda})^{-1} \times \sqrt{\Lambda}V^\top \mathrm{diag}(z)c. \tag{1.6}$$

We note that we slightly abuse the notation as the explicit description (1.6) is indeed the fist element (α-component) of the original definition (1.5).

Proposition 3.3 (Alternating best response). *Consider the set of update rules*

$$\begin{cases} \begin{bmatrix} \alpha_{t+1} \\ \beta_{t+1} \end{bmatrix} & = \mathrm{BR}(z_t) \\ z_{t+1} & = \mathbb{1}_n\big(\mathcal{J}(\alpha_t, \beta_t)\big), \end{cases} \tag{1.7}$$

where the set $\mathcal{J}$ and the function BR *are defined as in Lemma 3.2 and (1.5), respectively. Starting from an initialization (z_0, α_0, β_0), algorithm (1.7) converges after finitely many iterations to a limit cycle. If the set of this period behavior is singleton (i.e., the iterations convergence to a fixed point), then the variable z of the convergence point is the optimal solution of program $(\mathcal{M}_k)$.*

Proof. Recall that for any $z \in \{0,1\}^n$ the objective function L in (1.5) is strongly concave, and that admits a unique maximizer. On the other hand, the number of possible binary variable z is finite and bounded by 2^n. These two observations together then imply that the iterations (1.7) necessarily yield a period behavior with the cardinality at most 2^n. When the period is one, it then means that the best response algorithm has an equilibrium, implying that the min-max program $(\mathcal{M}_k)$ is indeed a minimax game with a Nash equilibrium. Namely, there exist $(\alpha^\star, \beta^\star) \in \mathbb{R}^k \times \mathbb{R}^m_+$ and $z^\star \in \{0,1\}^n$ such that for all $(\alpha, \beta) \in \mathbb{R}^k \times \mathbb{R}^m_+$, and $z \in \{0,1\}^n$, $\sum z_j \leq s$, we have

$$L(z^\star, \alpha, \beta) \leq L(z^\star, \alpha^\star, \beta^\star) \leq L(z, \alpha^\star, \beta^\star)$$

and, by definition, $z^\star$ solves the outer minimization of program $(\mathcal{M}_k)$. □

In the iterative scheme (1.7), evaluating the best response function $\mathrm{BR}(z_t)$ is equivalent to solving a linearly constrained convex quadratic program, which can be done efficiently using commercial solvers such as MOSEK [MOSEK ApS, 2019]. In case we have no linear constraints in the form of $Ax \leq b$, we can also use the explicit description (1.6). Therefore, the algorithm (1.7) is indeed highly tractable.

Remark 3.4 (Safe screening). *We expect that the periodic behavior anticipated by Proposition 3.3 typically has a periodicity larger than one. In fact, if the min-max characterization in Proposition 3.1 is not interchangeable without suffering from a duality gap, then the periodic behavior does have more than one element. In this setting, one can consider all the indices $\mathcal{J}(\alpha_t, \beta_t)$, where (α_t, β_t) belongs to the period behavior, as potential candidates for the ones of the optimal vector $z^\star$ in $(\mathcal{M}_k)$. This selection is indeed in accordance with the safe screening terminology of Atamturk and Gomez [2020].*

3.2 Dual program: a subgradient ascent approach

The second proposed algorithm aims to solve the dual of program ($\mathcal{M}_k$) described via

$$d_k^\star \triangleq \max_{\substack{\alpha\in\mathbb{R}^k \\ \beta\in\mathbb{R}^m_+}} \min_{\substack{z\in\{0,1\}^n \\ \sum z_j\leq s}} L(z,\alpha,\beta), \tag{$\mathcal{D}_k$}$$

where the function L was defined in (1.3). Thanks to the weak duality, it is obvious that $d_k^\star \leq J_k^\star$. The second approach is essentially the application of the subgradient ascent algorithm to the inner minimal function

$$f(\alpha,\beta) \triangleq \min_{z\in\{0,1\}^n, \sum z_j\leq s} L(z,\alpha,\beta). \tag{1.8}$$

We note that the continuous relaxation of the binary variable z from $\{0,1\}^n$ to $[0,1]^n$ in (1.8) does not change anything and the program remains equivalent to the original program ($\mathcal{D}_k$). Also notice that the function f in (1.8) is concave and piecewise quadratic, jointly in (α,β). This observation allows us to apply the classical subgradient algorithm from the convex optimization literature [Nesterov, 2003, Section 3.2.3].

Proposition 3.5 (Dual program)**.** *Consider the set of update rules defined as*

$$\begin{cases} \alpha_{t+1} = (1-\frac{1}{2}\kappa_t)\alpha_t - \frac{1}{2}\eta\kappa_t\sqrt{\Lambda}V^\top \mathrm{diag}(z_t)\times(c+V\sqrt{\Lambda}\alpha+A^\top\beta) \\ \beta_{t+1} = \max\Big\{0, \beta_t - \kappa_t b - \frac{1}{2}\eta\kappa_t A\mathrm{diag}(z_t)\times(c+V\sqrt{\Lambda}\alpha+A^\top\beta)\Big\} \\ z_{t+1} = \mathbb{1}_n\big(\mathcal{J}(\alpha_{t+1},\beta_{t+1})\big), \end{cases} \tag{1.9}$$

where $\{\kappa_t\}_t$ is the sequence of step sizes that satisfy the non-summable diminishing rule[2]

$$\lim_{t\to\infty}\kappa_t = 0, \quad \sum_{t=1}^{\infty}\kappa_t = \infty.$$

Then, algorithm (1.9) converges to the optimal value d_k^* of problem ($\mathcal{D}_k$), i.e.,

$$d_k^\star = \lim_{t\to\infty} f(\alpha_t,\beta_t).$$

Moreover, if the variable z_t also converges, then the convergent binary variable is the solution to program ($\mathcal{M}_k$) and the duality gap between ($\mathcal{M}_k$) and ($\mathcal{D}_k$) is zero, i.e., $d_k^\star = J_k^\star$.

Proof. Using standard results in variational analysis [Rockafellar and Wets, 2009], a subgradient of the function $f(\alpha,\beta)$ can be computed as

$$\frac{\partial f}{\partial\alpha} = -\frac{1}{2}\alpha - \frac{\eta}{2}\mathrm{diag}(\sqrt{\lambda})V^\top\mathrm{diag}(z_{\alpha,\beta})\times(c+V\mathrm{diag}(\sqrt{\lambda})\alpha+A^\top\beta),$$

[2] For instance, $\kappa_t = a/\sqrt{t}$ for a constant $a\in\mathbb{R}_+$.

$$\frac{\partial f}{\partial \beta} = -b - \frac{\eta}{2} A \mathrm{diag}(z_{\alpha,\beta})(c + V \mathrm{diag}(\sqrt{\lambda})\alpha + A^\top \beta),$$

where $z_{\alpha,\beta} = \mathbb{1}_n\big(\mathcal{J}(\alpha,\beta)\big)$ is the optimizer of the objective function $L(.,\alpha,\beta)$; see also (1.4). The subgradient algorithm updates the dual variables α_t, β_t by the following rule

$$\begin{bmatrix} \alpha_{t+1} \\ \beta_{t+1} \end{bmatrix} = \begin{bmatrix} \alpha_t \\ \beta_t \end{bmatrix} + \kappa_t \begin{bmatrix} \frac{\partial}{\partial \alpha} f(\alpha_t, \beta_t) \\ \frac{\partial}{\partial \beta} f(\alpha_t, \beta_t) \end{bmatrix},$$

where κ_t is the learning rate or stepsize. The computational complexity of the subgradient algorithm is well-known for concave and Lipschitz-continuous objective functions [Boyd et al., 2003, Nesterov, 2003]. In the remainder of the proof, we first verify that the classical results are applicable here too. To continue, let $\lambda = (\alpha, \beta)$ for short. Note that $z \in \{0,1\}^n$ which bounds the number of quadratic regions of f to 2^n which, in turn, implies that any level set of f is bounded. Since f is also concave and thus continuous, we conclude that f is in fact Lipschitz-continuous in any fixed level set of f. At iteration t, say $f(\lambda_t) = f_{z_t}(\lambda_t) := L(z_t, \alpha_t, \beta_t)$. This also means $g_t = \nabla f_{z_t}(\lambda_t)$ is a subgradient of f at λ_t. If t is large enough, then step size κ_t is small enough, and therefore the algorithm update $\lambda_{t+1} = \lambda_t + \kappa_t g_t$ increases the value of f_{z_t}. Hence, $f(\lambda_t) = f_{z_t}(\lambda_t) < f_{z_t}(\lambda_{t+1}) \leq f(\lambda_{t+1})$. Therefore, the subgradient algorithm remains afterwards within a level set of f, specified by $\{\lambda : f(\lambda) \geq f(\lambda_t)\}$ when t is sufficiently large. Now recall that f is Lipschitz-continuous in any fixed level set of f. So, we can apply the result of Boyd et al. [2003] for all sufficiently large $t \in \mathbb{N}$.

Concerning the second part of the assertion, suppose that the variable z_t also converges to a binary variable $z^\star$. Since the feasible set of the variable z is finite, the convergence assumption effectively implies that for all sufficiently large $t \in \mathbb{N}$, we have constant $z_t = z^\star$ in (1.9). As such, the dual algorithm (1.9) essentially reduces to

$$\begin{bmatrix} \alpha_{t+1} \\ \beta_{t+1} \end{bmatrix} = \begin{bmatrix} \alpha_t \\ \beta_t \end{bmatrix} + \kappa_t \begin{bmatrix} \frac{\partial}{\partial \alpha} L(z^\star, \alpha_t, \beta_t) \\ \frac{\partial}{\partial \beta} L(z^\star, \alpha_t, \beta_t) \end{bmatrix}.$$

The above argument allows us to also interpret the algorithm (1.9) as the subgradient ascent algorithm for the quadratic concave mapping $(\alpha, \beta) \mapsto L(z^\star, \alpha, \beta)$. This observation yields

$$\lim_{t\to\infty} L(z^\star, \alpha_t, \beta_t) = \max_{\alpha\in\mathbb{R}^k,\, \beta\in\mathbb{R}^m_+} L(z^\star, \alpha, \beta).$$

Thanks to the above equality, one can inspect that

$$J_k^\star \geq d_k^\star = \lim_{t\to\infty} L(z^\star, \alpha_t, \beta_t) = \max_{\substack{\alpha\in\mathbb{R}^k \\ \beta\in\mathbb{R}^m_+}} L(z^\star, \alpha, \beta) \geq J_k^\star,$$

where the first inequality is due to the weak duality between programs ($\mathcal{M}_k$) and ($\mathcal{D}_k$), and the last inequality follows from the definition of the optimal value of ($\mathcal{M}_k$). Since both sides of the above inequalities are $J_k^\star$, all the middle terms coincide. Thus, it concludes that $z^\star$ solves program ($\mathcal{M}_k$) and the zero duality gap holds, i.e., $J_k^\star = d_k^\star$. □

Similar to the best response algorithm in Proposition 3.3, we expect that in the long-run the duality algorithm (1.9) exhibits a period behavior over a number of $z_t \in \{0,1\}^n$. In this light, one can also consider the coordinates of ones elements of z_t as a safe screening suggestion (cf. Remark 3.4).

Remark 3.6 (Computational complexity)**.** *Formulating* ($\mathcal{P}_k$) *requires a PCA decomposition with a (crude) time complexity of* $\mathcal{O}(n^3)$*, and sorting for operation* (1.4) *takes* $\mathcal{O}(n \log n)$*. In addition, the best response algorithm in* (1.7) *has a complexity of* $\mathcal{O}(k^3 + nk)$*. Thus, overall the best response method has a complexity of* $\mathcal{O}(n^3 + t(k^3 + nk + n \log n))$*, where* t *are the number of iterations. The dual program algorithm* (1.9) *requires algebraic operations with complexity* $\mathcal{O}(nk)$*, sand that the overall complexity of the dual program is* $\mathcal{O}(n^3 + t(nk + n \log n))$*.*

3.3 Post-processing

Suppose, without any loss of generality, that either the best response algorithm (1.7) or the dual program algorithm (1.9) terminates after T iterations. By collecting the incumbent solutions $z_{T-p}, \ldots, z_T$ in the variable z over the last p iterations, we can form the unique indices

$$Z = z_{T-p} \mid z_{T-p+1} \mid \cdots \mid z_T \in \{0,1\}^n,$$

where $\mid$ represents the componentwise OR operator. Intuitively, the binary value of Z_i indicates if at least one of the last p incumbent solutions has the i-th element being non-zero. The vector Z thus represents the indices of x that are likely to be non-zero in the optimal solution of problem ($\mathcal{P}$). We now utilize the binary vector Z as an input to resolve the reduced problem

$$\begin{array}{ll} \min & \langle c, x\rangle + \langle x, Qx\rangle + \frac{1}{\eta}\|x\|_2^2 \\ \text{s.t.} & x \in \mathbb{R}^n, \\ & Ax \leq b,\ \|x\|_0 \leq s,\ |x| \leq MZ, \end{array} \tag{$\mathcal{P}_Z$}$$

where M is the big-M constant. If $Z_i = 0$, then the last constraint of ($\mathcal{P}_Z$) implies that $x_i = 0$ and a preprocessing step can remove this redundant component in x. As a consequence, the effective dimension of the variable x in ($\mathcal{P}_Z$) is upper bounded by the number of non-zero elements in Z, which is essentially $\|Z\|_0$. It is likely that Z has many elements that are 0, thus $\|Z\|_0 \ll n$ and problem ($\mathcal{P}_Z$) is easier to solve compared to ($\mathcal{P}$). In general, $\|Z\|_0 > s$, so ($\mathcal{P}_Z$) remains a

binary quadratic optimization problem. In the optimistic case when $\|Z\|_0 = s$, then the cardinality constraint $\|x\|_0 \leq s$ becomes redundant and $(\mathcal{P}_Z)$ reduces to a quadratic program.

In practice, the magnitude of M may affect the run time, and a tight value of M can significantly improve the numerical stability and reduce the solution time. We follow the suggestion from Hazimeh et al. [2020] to compute M as follows. Given the terminal solution z_T from either the best response algorithm (1.7) or the dual program algorithm (1.9), we solve problem $(\mathcal{P}_Z)$ with the input Z being replaced by z_T to get x_T. As z_T satisfies $\|z_T\|_0 = s$, this problem reduces to a quadratic program. To ensure that the big-M formulation adds no binding constraints we assign $M = 4\|x_T\|_\infty$.

Lastly, we emphasize that the solution z_t in the subgradient ascent algorithm (1.9) does not fluctuate significantly from one iteration to another. Thus, for the dual program approach, we need to set a periodic value p which is sufficiently large in order to recuperate meaningful signals on the indices. The best response method using the update (1.7), on the contrary, requires a smaller number of period p.

4 Numerical Experiments

We benchmark different approaches to solve problem $(\mathcal{P})$ in the sparse linear regression setting. All experiments are run on a laptop with Intel(R) Core(TM) i7-8750 CPU and 16GB RAM using MATLAB 2020b. The optimization problems are modeled using YALMIP [Löfberg, 2004], as the interface for the mixed-integer solver [MOSEK ApS, 2019]. Codes are available at: `https://github.com/RVreugdenhil/sparseQP`

We compare our algorithms against four state-of-the-art approaches. They include two screening methods: the safe `screening` method of Atamturk and Gomez [2020] and the `warm start` method in Bertsimas and van Parys [2017], and two direct optimization approaches: the Algorithm 7 in Beck and Hallak [2015] (denoted `BH Alg 7`) and the method of Yuan et al. [2020] (denoted `KDD`).[3] The best response alternation in Section 3.1 is referred to as `BR`, and the dual program approach in Section 3.2 is referred to as `DP`.

4.1 Empirical results for synthetic data

We generate the covariate $\omega \in \mathbb{R}^n$ and the univariate response $\xi \in \mathbb{R}$ using the linear model

$$\xi = x_{\text{true}}^\top \omega + \epsilon$$

[3] Available at `https://yuangzh.github.io/`

following the similar setup in Bertsimas and van Parys [2017]. The unobserved true vector $x_{\text{true}} \in \mathbb{R}^n$ has s-nonzero components at indices selected uniformly at random, without replacement. The nonzero components in x_{true} are selected uniformly at random from $\{\pm 1\}$. Moreover, the covariate ω is independently generated from a Gaussian distribution $\mathcal{N}(0, \Sigma)$, where Σ is parametrized by the correlation coefficient ρ as $\Sigma_{i,j} \triangleq \rho^{|i-j|}$ for all $i, j \in [n]$ and $0 \leq \rho \leq 1$. The noise ϵ is independently generated from a normal distribution $\mathcal{N}(0, \sigma^2)$ with

$$\sigma^2 = \frac{\text{var}(x_{\text{true}}^\top \omega)}{\text{SNR}} = \frac{x_{\text{true}}^\top \Sigma x_{\text{true}}}{\text{SNR}},$$

where SNR is a chosen signal-to-noise ratio [Xie and Deng, 2020].

To avoid a complicated terminating criterion, we run the `BR` method for $T_{\texttt{BR}} = 20$ iterations, and we run the `DP` method for $T_{\texttt{DP}} = 500$ iterations. We empirically observe that `BR` converges by $T_{\texttt{BR}} = 20$ on the synthetic data. The stepsize constant for `DP` (see Footnote 2) is set to $a = 4 \times 10^{-3}$. Regarding the post-processing step in Section 3.3, we fix $p_{\texttt{BR}} = 6$ and $p_{\texttt{DP}} = 50$ as the number of terminating solutions that are used to estimate the support Z. For the experiment on the synthetic data, the big-M constant is set to 4, because $\|x_{\text{true}}\|_\infty = 1$.

Our first experiment studies the impact of the regularization parameter η on the performance of `BR` and `DP` in terms of safe screening. To this end, we fix $N = 1000, n = 1000, s = 10$, $\rho = 0.5$ and $\text{SNR} = 6$ to generate the data.

The screening capacity of `BR` and `DP` is measured by the sparsity of the input parameter Z in problem $(\mathcal{P}_Z)$, this reduced size is measured by $\|Z\|_0$. A similar quantity can be computed for `screening`. We choose the dimension of subspace $k = 400$ to ensure that our the principal component approximation generates a good quality solution to the original problem. Table 1.1 reports the screening capacity and we can observe that `screening` effectively reduces the dimension for small values of η, which is in agreement with the empirical results reported in Atamturk and Gomez [2020]. However, `screening` performs less convincingly for $\eta \geq 1$. Our methods `BR` and `DP` perform more consistently over the whole range of η: they can reduce $(\mathcal{P}_Z)$ to a quadratic program for 59% and 81% of all instances respectively.

We also study the performance of our methods in a setting with $\rho \geq 0.7$. We measure the quality of the estimator x by the mean squared error on the data

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} \|\xi_i - x^\top \omega_i\|_2^2.$$

We fix the regularisation term $\eta = 10$. Setting η to a lower value would cause unwanted shrinkage of the estimator x, which increases the MSE: instances where $\eta = 0.1$ reported a MSE for all methods of at least 5 times larger than that of the MSE using $\eta = 10$. As seen in Table 1.1 for this specific

Table 1.1: Effective problem size measured by $\|Z\|_0$ for varying η, averaged over 25 replications. Lower is better. Values are rounded to nearest integer, asterisks denote that $\|Z\|_0 = s$ on all instances.

	DP $k = 400$	BR $k = 400$	screening
$\eta = 10^2$	69	20	1000
$\eta = 10$	$10^\star$	20	1000
$\eta = 1$	10	10	809
$\eta = 10^{-1}$	$10^\star$	10	463
$\eta = 10^{-2}$	$10^\star$	$10^\star$	45
$\eta = 10^{-3}$	$10^\star$	$10^\star$	10

η the screening method does not reduce the problem size, so we compare to the MSE generated by warm start.

We observe in Table 1.2 that DP outperforms the other methods in terms of MSE with highly correlative data ($\rho \geq 0.8$). A possible explanation for why DP outperforms the warm start is that the eigenvalue decomposition can convert highly correlative features to independent principal components [Liu et al., 2003].

Table 1.2: MSE over different ρ averaged over 25 independent replications. Lower is better.

	DP $k = 400$	BR $k = 400$	warm start	BH Alg 7	KDD
$\rho = 0.7$	1.829	1.829	1.829	1.897	1.829
$\rho = 0.8$	1.863	1.969	1.988	2.114	1.866
$\rho = 0.9$	1.895	3.666	3.714	2.570	1.984

4.2 Empirical results for real data

We benchmark different methods using real data sets [Dua and Graff, 2017]; the details are listed in Table 1.3. We preprocess the data by normalizing each covariate and target response independently to values in the range $[0, 1]$. For each independent replication, we randomly sample 70% of the data as training data and 30% as test data. The training set and the test set have N_{train} and N_{test} samples, respectively.

The number of iterations for the BR and DP are chosen to be $T_{\text{BR}} = 40$ and $T_{\text{DP}} = 5000$, respectively. The number of terminating solutions that are used to estimate the support Z in the postprocessing

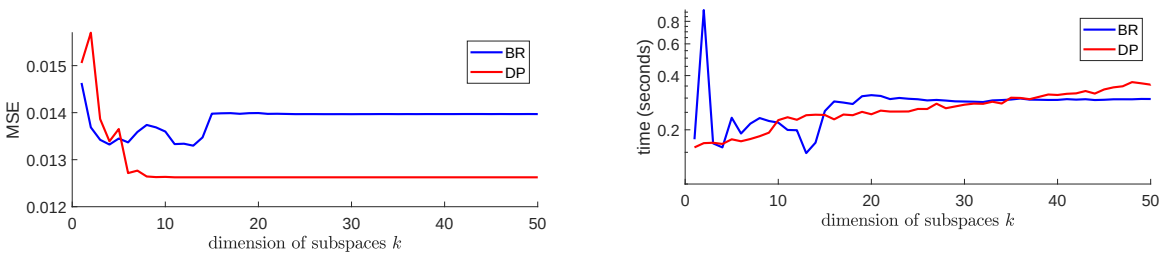
Table 1.3: List of UCI datasets used for experiments alongside their feature size n, total sample size $\bar{N}$ and the dimension of subspace $\hat{k}$, specified earlier in Figure 1.2.

Name	n	$\bar{N}$	$\hat{k}$
Facebook (FB)	52	199,030	14
OnlineNews (CR)	58	39,644	20
SuperConductivity (SC)	81	21,263	9
Crime (CR)	100	1,994	53
UJIndoor (UJ)	465	19,937	49

step is fixed to $p_{\texttt{BR}} = 10$ and $p_{\texttt{DP}} = 100$ for the two methods. The stepsize constant (see Footnote 2) is set $a = 2 \times 10^{-3}$. Moreover, we fix the big-M constant using the procedure described in Section 3.3. For the real datasets, the number of samples $\bar{N}$ is sufficiently bigger than the dimension n, thus we set the ridge regularization parameter to $\eta = \sqrt{N_{\text{train}}}$ so that the effect of regularization diminishes as $\bar{N}$ increases. We choose the sparsity level $s = 10$, similar to Section 4.1, and set the time limit for MOSEK to 300 seconds.

Figure 1.4 illustrates that the MSE of `DP` monotonically decreases with the dimension of subspace for $k < 8$ and plateaus for $k \geq 8$. The MSE of `BR` is non monotonic and converges when $k \geq 15$ even though the minimum is achieved at $k = 13$. We define $k^\star$ as the (minimal) dimension of subspace k corresponding to the lowest MSE. For the (SC) dataset, $k^\star_{\texttt{BR}} = 13$ and $k^\star_{\texttt{DP}} = 15$. Coincidentally, we observe that $k^\star$ is close to the value $\hat{k}$ reported in Table 1.3. This observation also persists empirically for the other datasets.

Table 1.4 shows that `DP` delivers a lower in-sample MSE than `BR` in 4 out of 5 datasets, and `DP` also has a lower in-sample MSE than the `warm start`, `BH Alg 7` and `KDD` for all datasets. The `warm`



(a) in-sample Mean Squared Error (b) computational time

Figure 1.4: Effects of the dimension of subspaces k on performance of `BR` and `DP` on the Superconductivity dataset. Results are averaged over 50 independent train-test splits.

Table 1.4: In-sample MSE on real datasets, averaged over 50 independent train-test splits. Lowest error for each case is highlighted in grey.

	DP $k = 40$	DP $k = \hat{k}$	BR $k = 40$	BR $k = \hat{k}$	warm start	screening	BH Alg 7	KDD
(FB)	3.039×10^{-4}	3.040×10^{-4}	3.036×10^{-4}	$\mathbf{3.034\times10^{-4}}$	out of memory	3.036×10^{-4}	3.220×10^{-4}	3.436×10^{-4}
(ON)	$\mathbf{1.912\times10^{-4}}$	1.913×10^{-4}	1.914×10^{-4}	1.914×10^{-4}	1.914×10^{-4}	1.914×10^{-4}	1.921×10^{-4}	1.922×10^{-4}
(SC)	1.262×10^{-2}	1.263×10^{-2}	1.396×10^{-2}	1.368×10^{-2}	1.326×10^{-2}	$\mathbf{1.256\times10^{-2}}$	1.455×10^{-2}	1.470×10^{-2}
(CR)	2.775×10^{-2}	2.775×10^{-2}	2.807×10^{-2}	2.801×10^{-2}	2.800×10^{-2}	$\mathbf{2.760\times10^{-2}}$	3.072×10^{-2}	3.063×10^{-2}
(UJ)	$\mathbf{2.118\times10^{-2}}$	2.294×10^{-2}	2.673×10^{-2}	2.678×10^{-2}	2.440×10^{-2}	2.267×10^{-2}	3.804×10^{-2}	3.066×10^{-2}

`start` method runs out of memory for the (FB) dataset because it requires storing and computing based on a kernel matrix $K = [\omega_i^\top \omega_j]_{i,j}$ of dimension $N \times N$. This is in stark contrast to our proposed approach that computes only a matrix Q of dimension $n \times n$, and then further reduce the computational burden by truncating the SVD of Q. The memory usage of our method is hence not sensitive to the number of samples N.

The `screening` method outperforms on the (SC) and (CR) dataset, however a careful examination of Table 1.5 shows that `screening` does minimal reduction effects for these two datasets. The result of `screening` in Table 1.4 on the (SC) and (CR) datasets is essentially the results obtained by applying the MOSEK solver to the original problem (reaching a time limit of 300 seconds).

Table 1.5 also shows that our `DP` and `BR` methods can effectively reduce the number of effective variables. Our methods deliver a solution $x^\star$ in around 1 second for all datasets, including the time spent on computing the eigendecomposition of Q and the solution time for solving ($\mathcal{P}_Z$) using MOSEK.

Table 1.5: Reduced problem size over different data rounded average over 50 independent train-test splits. Lower is better.

	DP $k = \hat{k}$	BR $k = \hat{k}$	screening
(FB)	11	20	49
(ON)	15	20	57
(SC)	11	20	77
(CR)	12	20	100
(UJ)	16	20	465

Remark 4.1 (Choice between `DP` and `BR`)**.** *We have no theoretical or consistent numerical justification in favor of one of the proposed algorithms* `DP` *or* `BR`*. However, Algorithm* `BR` *in Proposition 3.3 typically converges faster (Fig. A in supplementary) while Algorithm* `DP` *offers a better solution (Fig. 1.4a and Tables 1.2, 1.5). We thus suggest* `DP` *and* `BR` *as complementary approaches to solve the problem.*

Chapter 2

Sparse inductive matrix completion using updated side information

Abstract

We propose a novel approach for solving the problem of matrix completion with side information by effectively rewriting it to a sparse quadratic programming problem. Solving this program gives us a selection of features in the side information matrix, after which the matrix completion problem can be solved using convex operations. To overcome noisy or weakly informative features in the side information matrix, we propose to update the side information matrix by replacing the non selected features. Numerical experiments show that our method is highly scalable and competitive with current state-of-the-art matrix completion methods in real and synthetic data settings.

1 Introduction

Matrix completion has been a basis of many machine learning approaches for computer vision [Chen and Suter, 2004], recommender systems [Rennie and Srebro, 2005, Sindhwani et al., 2010], signal processing [Ning and Karypis, 2012, Weng and Wang, 2012], and among many others. Classically, low-rank matrix completion methods are based on matrix decomposition techniques which require only the partially observed data in the matrix [Candès and Recht, 2012, Keshavan et al., 2010]. Standard matrix completion tries to recover the low-rank matrix by solving the following problem

[Candes and Plan, 2010, Keshavan et al., 2010]:

$$\begin{aligned} \min_{X} \quad & \frac{1}{nm} \sum_{(i,j)\in\Omega} (X_{ij} - A_{ij})^2 \\ \text{s.t.} \quad & \text{Rank}(X) = s \end{aligned} \tag{2.1}$$

where $\mathbf{A} \in \mathbb{R}^{n\times m}$ is the partially observed low-rank matrix that needs to be recovered, $\Omega \subseteq \{1, \cdots, n\} \times \{1, \cdots, m\}$ is the set of indices where the corresponding components in $\mathbf{A}$ are observed, the mapping $R_\Omega(\mathbf{A}) : \mathbb{R}^{n\times m} \to \mathbb{R}^{n\times m}$ gives another matrix whose (i, j) -th entry is $\mathbf{A}_{i,j}$ if $(i, j) \in \Omega$ (or 0 otherwise). The rank constraint in 2.1 can be equivalently formulated as the existence of two matrices $U \in \mathbb{R}^{n\times s}, V \in \mathbb{R}^{m\times s}$ such that $X = UV^T$. Therefore, the problem can be restated as:

$$\begin{aligned} \min_{X,U,V} \quad & \frac{1}{nm} \sum_{(i,j)\in\Omega} (X_{ij} - A_{ij})^2 \\ \text{s.t.} \quad & X = UV^T \end{aligned} \tag{2.2}$$

The most straightforward way of solving 2.2 is by alternating minimization introduced by Jain et al. [2012]. In this method either U or V is fixed and each subproblem can be written as a standard least squares problem. Due to its simplicity, low memory requirements and flexibility alternating minimization is widely used in the literature [Escalante and Raydan, 2011]. Often in addition to the partially observed matrix there exists information about row- or column- objects. For example, side-channel information in the form of user profiles and movie genres is natural in recommender systems and was proved to be useful in real-world applications [Hannon et al., 2010]. We focus on the one-sided information case, where there only exists information about row- or column- objects. This case is more relevant as features related to users are often fragmented and increasingly constrained by data privacy regulations, while features about products are easy to obtain [Bertsimas and Li, 2020b]. We denote the side information matrix as B where $V \subseteq B$.

The problem we consider here is that for every column $j = 1, \ldots, m$, we have a given p dimensional feature vector B_j with $p \geq s$ that contains the information we have on column j. Given side data matrix $B \in \mathbb{R}^{m\times p}$ we postulate that $X = UB^T$, where $U \in \mathbb{R}^{n\times p}$ is the matrix of feature exposures. To ensure the rank condition we enforce an ℓ_0 constraint on U [Miller, 2002]. We take the ℓ_0 norm over the infinity norm $\|U\|_\infty$ and denote this by $\|U\|_{\infty,0}$, which controls the number of nonzero columns in U. The matrix completion problem with side data B can be written as:

$$\begin{aligned} \min_{X,U} \quad & \frac{1}{nm} \sum_{(i,j)\in\Omega} (X_{ij} - A_{ij})^2 \\ \text{s.t.} \quad & X = UB^T, \|U\|_{\infty,0} \leq s \end{aligned} \tag{2.3}$$

Problem (2.3) is similar to the interpretable matrix completion problem described in Bertsimas and Li [2020b].

Contributions. We summarize the contributions as follows.

(i) **Novel reformulation:** We present a novel reformulation of problem (2.3). The resulting equation only requires solving a single sparse quadratic programming problem of dimension p. The solution gives the indices of the non zero columns in U in (2.3) and the elements in U can be computed using convex operations.

(ii) **Updating the side information:** We propose an iterative method to update the features in the side information matrix B which correspond with the zero columns in U. This can improve the approximation of the partially observed matrix A when $\text{row}(A) \nsubseteq \text{col}(B)$.

(iii) **Numerical comparison** We present computational results on both synthetic and real datasets that show that our method matches or outperforms current state-of-the-art methods in terms of both scalability and accuracy.

2 Literary review

Inductive matrix completion. The works by Xu et al. [2013], Dhillon et al. [2013] introduced the concept of using side information under the name of inductive matrix completion. The work of Xu et al. [2013] showed that with perfect side information (i.e. $\text{row}(A) \subseteq \text{col}(B)$) we only need $O(\log(n))$ samples to retrieve the full matrix, where non inductive methods require $O(ns\log^2(n))$ samples [Candès and Recht, 2012, Candes and Tao, 2010, Recht, 2011]. Until recently most works on inductive matrix completion were based on an alternating optimization approach [Xu et al., 2013, Dhillon et al., 2013]. Nevertheless, these methods often cannot recover the ground truth matrix exactly [Zhang et al., 2018]. To overcome this Tu et al. [2016], Zhao et al. [2015], Zhang et al. [2018] propose first-order algorithms to solve low-rank matrix estimation problems, where they use (projected) gradient descent to refine the initial solution. In the case of non perfect side information Chiang et al. [2015] proposes to add a low rank matrix N to X such that $\text{rank}(X+N) = s$, where N is outside the features space of B. Moreover Si et al. [2016] proposes a nonlinear mapping on B to guarantee $\text{row}(A) \subseteq \text{col}(B)$, however this approach is only applicable in a low dimensional regimes i.e. $n \leq 10,000$.

Sparse inductive matrix completion. Sparse inductive matrix completion allows us to efficiently filter out non-informative side-channel features [Nazarov et al., 2018]. This has several benefits like: minimizing the risk of overfitting, lowering computational and storage complexity

Table 2.1: How our method stands in the current literature

	our method	alternating minimization	Bertsimas and Li [2020a]	Zhang et al. [2018]
inductive	✔	✖	✔	✔
update B	✔ update 'worst' elements in B	✔ update 'best' elements in B	✖	✖
include sparsity	✔	✖	✔	✖

which is desirable in some applications [Acar et al., 2012] and guaranteeing the latent features to be interpretable (as long as the original features are). Common convex heuristic approaches e.g., Lasso (ℓ_1) [Tibshirani, 1996] and Elastic Net ($\ell_1 - \ell_2$) [Zou and Hastie, 2005] are applied in matrix completion literature to include ℓ_0 constraints, [Nazarov et al., 2018, Lu et al., 2016, Soni et al., 2016]. Despite their scalability, convex heuristic approaches are inherently biased since the ℓ_1-norm penalizes large and small coefficients uniformly. In contrast, solvers that directly tackle sparse regression do not suffer from unwanted shrinkage [Bertsimas and van Parys, 2017, Hazimeh et al., 2020, Dedieu et al., 2020, Gomez and Prokopyev, 2018, Vreugdenhil et al., 2021].

Recent papers have adopted the exact formulation by Bertsimas and van Parys [2017] and applied it to matrix completion problems. The method of Bertsimas and Li [2020b] is aimed to select exactly s features from the known features in the side information and call this interpretable matrix completion, while Bertsimas and Li [2020a] allows the factorization to be any linear combination of the features in the side information.

Given the existing literature mentioned above, to our best of knowledge, our approach is the first to rewrite $\mathcal{M}$ to a sparse quadratic programming problem and to update features in the B matrix in high dimensional regimes.

3 Rewriting the matrix completion problem

We add a ridge regularization term $\|U\|_2^2$ [Hoerl and Kennard, 1970] to the original problem 2.3 to reduce the effects of noise and/or uncertainty [Ghaoui and Lebret, 1997, Mazumder et al., 2020]. Throughout the paper $\|\cdot\|_2$ is the Frobenius norm. Specifically, we address problem ($\mathcal{M}$)

$$\begin{aligned} \min_{U} \quad & \frac{1}{nm}\Big(\sum_{(i,j)\in\Omega}(X_{ij}-A_{ij})^2+\frac{1}{\eta}\|U\|_2^2\Big) \\ \text{s.t.} \quad & X = UB^T, \|U\|_{\infty,0} \leq s \end{aligned} \tag{$\mathcal{M}$}$$

where $\eta > 0$ is a given parameter that controls the strength of the regularization term. We can rewrite problem ($\mathcal{M}$) to a summation of quadratic programming problems using a similar approach to Bertsimas and Li [2020a] which rewrites ($\mathcal{M}$) to a summation of sparse regression problems.

Proposition 3.1. *[Quadratic programming reformulation] Problem ($\mathcal{M}$) can be reformulated as a summation of quadratic programming problems:*

$$\mathcal{J}^\star = \min_{\substack{U \\ \text{s.t.} \|U\|_{\infty,0} \leq s}} \frac{1}{nm} \sum_{i=1}^{n} u_i(Q_i)u_i^\top - 2(c_i)^\top u_i^\top + \frac{1}{\eta}\|u_i\|_2^2 \tag{$\mathcal{P}$}$$

where $U^\top = [u_1, \ldots, u_n]$ and u_i represents the i^{th} row in matrix U, $Q_i = B^\top W_i B, c_i = a_i W_i B$ and $W_1, \cdots, W_n \in \mathbb{R}^{m\times m}$ are diagonal matrices:

$$(W_i)_{jj} = \begin{cases} 1, & (i,j) \in \Omega \\ 0, & (i,j) \notin \Omega \end{cases}$$

To reduce the complexity of generating $B^\top W_i B$ and $a_i W_i B$ we follow the work by Bertsimas and Li [2020a]. Let us denote m_i as the number of non-zero entries of W_i. This is the number of known entries per row. Then define $B_{W_i} \in \mathbb{R}^{m_i \times p}$ as the matrix of $\boldsymbol{W}_i\boldsymbol{B}$ after removing the all-zero columns. Note that B_{W_i} can be created efficiently through subsetting, and its creation does not impact the asymptotic running time. Then similarly, we denote $\boldsymbol{a}_{W_i} \in \mathbb{R}^{1\times m_i}$ as $\boldsymbol{a}\boldsymbol{W}_i$ with all the zero elements removed.

$$B^\top W_i B = B_{W_i}^\top B_{W_i} \quad a_i W_i B = a_{W_i} B_{W_i}$$

Note that ($\mathcal{P}$) is equivalent to n convex quadratic programming problems when the indices of the nonzero columns in U are given. In an attempt to find these indices we generalize ($\mathcal{P}$) to a single sparse quadratic programming problem. We denote the generalization over all vectors $u_1, \ldots, u_n$ as $\bar{u} \in \mathbb{R}^p$

$$\begin{aligned} \min_{\bar{u}} \quad & \bar{u} \sum_i Q_i \bar{u}^\top + \sum_i c_i^\top \bar{u}^\top + \frac{1}{\eta}\|\bar{u}\|_2^2 \\ \text{s.t.} \quad & \bar{u}_j(1 - z_j) = 0, \quad \forall i = 1 \ldots p \\ & z \in \{0,1\}^p, \|z\|_0 \leq s \end{aligned} \tag{$\mathcal{P}_z$}$$

Using the resulting binary vector z we define $B_Z \in \mathbb{R}^{m\times s}$ as the matrix of $B\text{diag}(z)$ after removing the all-zero columns. We can determine U using $V = B_Z$

$$u_i = a_{W_i} V_{W_i} (V_{W_i}^\top V_{W_i} + \mathbb{I}_s/\eta)^{-1}$$

3.1 Solving $\mathcal{P}_z$

Note that we can use any sparse quadratic programming method to solve ($\mathcal{P}_z$). Quadratic mixed-integer optimization problems have received relatively little attention in the literature [Bertsimas and Shioda, 2009]. Recently several works have attempted to solve problems like ($\mathcal{P}_z$) in high dimensional settings [Xie and Deng, 2020, Vreugdenhil et al., 2021, Beck and Hallak, 2015]. In this perspective we compare two state of the art methods. The first method is a modified version of the approach by Vreugdenhil et al. [2021], where we change the diminishing stepsize to a backtracking stepsize [Fliege and Svaiter, 2000] for the gradient ascent. Regarding convergence, we can apply the result of Gordon and Tibshirani [2012]. Numerical examples comparing the convergence of the algorithm using both a diminishing and backtracking stepsize displays that we can achieve faster convergence using the backtracking stepsize.

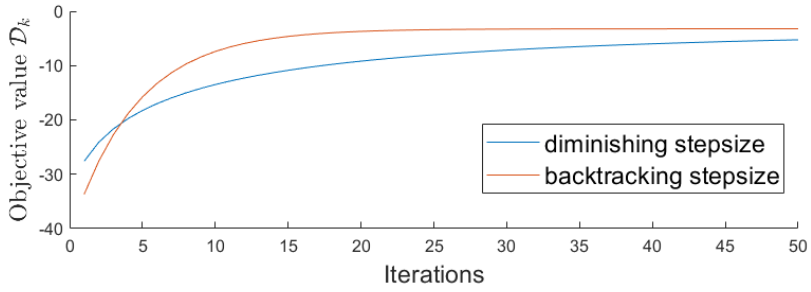


Figure 2.1: Convergence of the diminishing stepsize and backtracking stepsize

The second approach is the method by Xie and Deng [2020], which uses a forward greedy approach. The method by Vreugdenhil et al. [2021] and Xie and Deng [2020] have comparable MAPE and computational time for the synthetic data setting described by (2.4). This result can be found in the supplementary B. However in a sparse regression setting similar to Xie and Deng [2020] we observe that for larger s the method by Vreugdenhil et al. [2021] outperforms the method by Xie and Deng [2020]. Based on the observation in Table 2.2 we select the method by Vreugdenhil et al. [2021] to solve ($\mathcal{P}_z$). We denote rewriting ($\mathcal{M}$) to ($\mathcal{P}_z$) and solving it as sparse quadratic programming for matrix completion (`SQPMC`).

Table 2.2: Mean squared error for different s averaged over 20 different replications. Lowest error for each case is highlighted in grey

	$s = 10$	$s = 20$	$s = 30$	$s = 40$	$s = 50$
Xie and Deng [2020]	1.917	3.705	5.566	7.773	10.022
Vreugdenhil et al. [2021]	1.917	3.573	5.269	7.350	9.460

4 Imperfect features

To be less reliant on the quality of the side information in cases where $\text{row}(A) \nsubseteq \text{col}(B_Z)$ we propose to update B. We denote the columns in B corresponding to the zero columns in U by $B_Y \in \mathbb{R}^{m\times p-s}$. In an iterative scheme we can update the columns in B_Y, where it is important to note that optimizing needs not to be restrictive to rank s in early stages. For example in the case where Ω is the set of all indices choosing the initial rank as m is actually the optimal choice in the first step so at each time step we define a rank s_t. The update rule replaces s_t randomly selected columns in B_Y with the first s_t right-eigenvectors of the SVD of the approximation matrix X. In the case where $s_t > p - s_t$ we only replace $p - s_t$ columns in B_Y with the first $p - s_t$ right-eigenvectors. We also highlight that an alternating minimization approach essentially updates the column in B corresponding to the nonzero entries in U.

5 Numerical experiments

We benchmark different approaches to solve problem ($\mathcal{M}$). We compare our method to the `MPPF` method in Zhang et al. [2018], which uses an alternating gradient descent approach to inductive matrix completion, the `fastImpute` method in Bertsimas and Li [2020a], which uses a sparse inductive matrix completion approach and the classical alternating minimization (denoted `Alt`) method described by Jain et al. [2012]. The `MPPF` and `fastImpute` method are downloaded from the github pages of the authors `MPPF` (Matlab 2020b) `fastImpute` (Julia 1.5.3). All experiments are run on a laptop with Intel(R) Core(TM) i7-8750 CPU and 16GB RAM using MATLAB 2021a.

5.1 Empirical results for synthetic data

We test on two different synthetic data settings. First we generate matrix $A \in \mathbb{R}^{n\times m}$ according to the following formula:

$$\begin{aligned} A &= UB^\top + E \\ \text{s.t.} \quad &\|U\|_{\infty,0} \leq s \end{aligned} \tag{2.4}$$

$U \in \mathbb{R}^{n\times p}$ has s-nonzero columns at indices selected uniformly at random without replacement. The nonzero component in U and all elements in $B \in \mathbb{R}^{m\times p}$ are drawn elementwise from a uniform distribution of $[0, 1]$ and the noise matrix E is drawn elementwise from $\mathcal{N}(0, 0.01)$. We randomly select a fraction $\mu \in [0, 1]$ to be missing in A. For the synthetic dataset we denote the performance

Table 2.3: MAPE and computational time over different parameters averaged over 20 independent replications. Lower is better.

						SQPMC		fastImpute	
	n	m	p	μ	s	MAPE	time (s)	MAPE	time(s)
	10^3	10^3	10^2	0.95	10	0.39	0.18	3.00	1.77
n	10^4	10^3	10^2	0.95	10	0.39	2.09	2.80	5.62
	10^5	10^3	10^2	0.95	10	0.39	19.19	2.52	24.74
m	10^3	10^4	10^2	0.95	10	0.35	1.52	2.82	3.24
	10^3	10^5	10^2	0.95	10	0.35	20.46	2.79	9.27
p	10^3	10^3	5×10^2	0.95	10	0.39	1.74	3.55	2.82
	10^3	10^3	10^3	0.95	10	0.39	5.87	4.09	3.80
s	10^3	10^3	10^2	0.95	5	0.82	0.19	6.11	0.60
	10^3	10^3	10^2	0.95	30	0.19	0.22	5.80	89.48
μ	10^3	10^3	10^2	0.8	10	0.36	0.36	2.71	1.00
	10^3	10^3	10^2	0.99	10	2.91	0.12	21.28	2.06

of the different methods with the Mean Absolute Percentage Error (MAPE):

$$\text{MAPE} = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{|X_{ij} - A_{ij}|}{|A_{ij}|}$$

To demonstrate the scalability of `SQPMC` we compare it to `fastImpute` for different n, m, p, μ and s and we set the regularization term to $\eta = 10^2$. In Table 2.3 we observe that `SQPMC` can achieve a lower MAPE for all instances compared to `fastImpute` and `SQPMC` is faster for all instances except two compared to `fastImpute`. The running time of our method is more susceptive to the number of columns m and features p, whereas the `fastImpute` is more susceptive to the sparsity level s.

The second synthetic data setting allows the matrix A to be the factorization of any linear combination of the features in the side information, similar to Bertsimas and Li [2020a]

$$A = UC^\top B^\top + E$$

where $U \in \mathbb{R}^{n \times s}, C \in \mathbb{R}^{p \times s}, B \in \mathbb{R}^{m \times p}$. The matrices U, C, B are drawn elementwise from a uniform distribution of $[0, 1]$ and the noise matrix E is drawn elementwise from $\mathcal{N}(0, 1)$. In this data setting we can use `SQPMC` to update the side information B in an iterative algorithm as explained in Section 4. We denote $S = [s_1, \dots s_T]$ as the vector containing the different sparsity levels we use to update B, where $s_T = s$. S is generated using linearly spaced values starting at s_1

Table 2.4: MAPE and computational time over different parameters averaged over 20 independent replications. Lower is better.

					SQPMC		update B	
n	m	p	mu	s	MAPE	time (s)	MAPE	time(s)
10^3	10^3	10^2	0.95	15	10.56	0.27	2.69	2.60
10^4	10^3	5×10^2	0.95	50	14.14	19.39	2.66	364.47
10^3	10^4	2×10^2	0.95	40	6.24	4.13	0.98	46.17
10^4	10^4	80	0.99	4	21.49	6.36	2.64	85.54

and going to s in T steps. We set $s_1 = 0.9p$ and $T = 3$, the remaining variables are the same as used in the first synthetic dataset.

In Table 2.4 we observe that updating B results in a lower MAPE for all instances on this particular synthetic data setting. However the computational time using updates in B does increase.

5.2 Emperical results for real data

We run experiments on the MovieLens dataset [1][Harper and Konstan, 2015]. The MovieLens dataset aims to predict ratings of users on unseen movies. The total matrix A has 162,541 rows and 13,816 columns. The side information B consists out of 1,139 features. The features in the side information data are made up by the tag genome. The tag genome encodes how strongly movies exhibit particular properties represented by tags (atmospheric, thought-provoking, realistic, etc.) as described in Vig et al. [2012]. Moreover every element in the tag genome lies in $[0, 1]$. Out of all features in the side information we only select the columns B_j with mean value greater than 0.1, because we observe that the features with lower mean value are almost never part of B_Z. To display the scalability of `SQPMC` we create three different datasets. We take a relatively low number of training data to show the performance of our method in regimes with a low number of observed

[1]MovieLens: https://grouplens.org/datasets/movielens/25m/

Table 2.5: Sizes of the different MovieLens datasets

dataset	n	m	p	s	μ	train/test
M_1	10,000	5,000	497	30	0.9798	0.4/0.6
M_2	20,000	10,000	505	40	0.9867	0.3/0.7
M_3	50,000	13,816	503	50	0.9890	0.2/0.8

Table 2.6: Performance on training and testing data of different methods on the MovieLens dataset

Dataset	Method	train	test	time
M_1	`SQPMC`	17.83	27.69	21.33
	`SQPMC + update B`	17.77	27.39	151.01
	`MPPF`	35.61	36.07	936.94
	`fastImpute`	18.08	33.28	410.04
	`Alt`	19.24	27.45	23.90
M_2	`SQPMC`	16.39	28.36	55.24
	`SQPMC + update B`	16.13	28.12	524.38
	`MPPF`	33.68	34.24	3516.95
	`fastImpute`	19.68	33.09	1135.34
	`Alt`	17.41	29.22	90.79
M_3	`SQPMC`	13.68	29.35	153.21
	`SQPMC + update B`	13.79	28.93	1882.64
	`MPPF`	Exceed time limit		
	`fastImpute`	Out of memory		
	`Alt`	15.08	31.72	712.79

components in A.

We set $\eta = 10$ for `SQPMC` and `MPPF`, because the `fastImpute` and `Alt` method are more prone to overfitting we set $\eta = 1$ for these methods. To further reduce the effects of overfitting for the `Alt` method we take $s_{\texttt{Alt}} = s/3$. Furthermore regarding the sparsity levels for the update on B we set $S = [4s, 2.5s, s]$.

In table 2.6 we observe that for all sizes of the MovieLens dataset, `SQPMC` can achieve a lower MAPE on the training and testing data than `fastImpute`, `MPPF` and `Alt` with and without updating B. We also observe that using updated side information matrix can slightly decrease the MAPE on the testing data and gives comparable MAPE on the training data. This does come with an increase in computational time. Table 2.6 also shows that `SQPMC` without updating B has a lower computational time for all instances compared to the other methods.

Appendix A

Principal component hierarchy for sparse quadratic programs

A Additional Numerical Experiments

A.1 Comparison of Computational Time to `warm start`

We study the impact of the sample size N on the recovery quality of the solution. We fix $n = 1000$, $s = 10$, $\rho = 0.5$, SNR $= 6$ and $\eta = 10$. We showcase the computational time of our methods and of the `warm start` in Figure A.1, the computational time is defined as the time needed to generate $x^\star$. Note that the `BR` method uses MOSEK to obtain the solution to $x^\star$ because it does not converge to a single set z for $\eta = 10$, so the solver time is also included in the computational time. We run the `BR` method for $T_{\texttt{BR}} = 20$ iterations, and we run the `DP` method for $T_{\texttt{DP}} = 500$ iterations.

We observe that the computational time of the `DP` method increases monotonically with the sample size N. Note that $T_{\texttt{BR}} \ll T_{\texttt{DP}}$ so calculating $Z_{\texttt{BR}}$ requires less time than $Z_{\texttt{DP}}$. We observe that when $N = 100$ the `BR` and the `warm start` have a higher computational time than for $N = 500$. For the `BR`, this is because the number of non-zero elements in Z (i.e., $\|Z\|_0$) is larger for $N = 100$ than for $N = 500$, hence MOSEK takes more time for $N = 100$. The MSE of all methods is similar when $N \geq 500$, when $N = 100$ the MSE of all methods differs significantly at every instance. This is also observed by Bertsimas and van Parys [2017], which states that the computational time and MSE deteriorate as N gets smaller relative to n.

We observe that `BR` and `DP` perform particularly well in terms of computational time in ranges where $N > n$ compared to the `warm start`. The running time of our method is less susceptive to the number of samples N. This is in stark contrast to the `warm start`, in which the kernel matrix

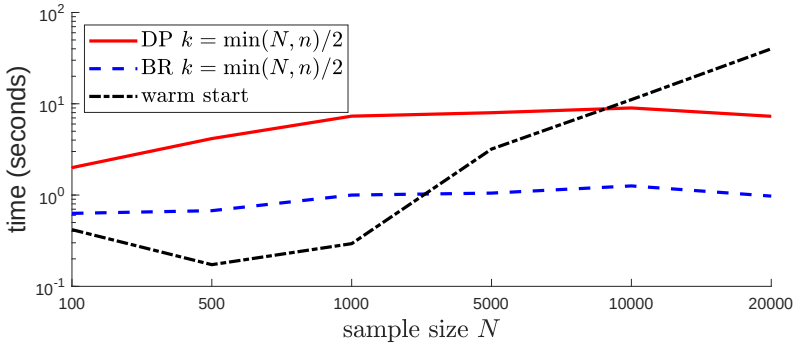


Figure A.1: Computational time over different sample sizes averaged over 25 replications

of dimension N-by-N is stored.

A.2 Comparison for Different SNR and sparsity level

We extend the comparison made in the paper for different values of SNR and s.

Table A.1: MSE over different SNR averaged over 25 independent replications. Lower is better.

	DP $k = 400$	BR $k = 400$	warm start	Beck Alg 7	KDD
SNR $= 20$	0.588	0.588	0.588	0.588	0.588
SNR $= 6$	1.767	1.767	1.767	1.767	1.767
SNR $= 3$	3.452	3.452	3.452	3.452	3.452
SNR $= 1$	10.190	10.190	10.190	10.198	10.205
SNR $= 0.05$	194.592	194.561	194.560	194.756	199.928

Table A.2: MSE over different s averaged over 25 independent replications. Lower is better.

	DP $k = 400$	BR $k = 400$	warm start	Beck Alg 7	KDD
$s = 5$	0.887	0.887	0.887	0.887	0.887
$s = 10$	1.767	1.767	1.767	1.767	1.767
$s = 20$	3.435	3.435	3.435	3.557	3.450
$s = 30$	5.050	5.050	5.058	5.888	5.440
$s = 40$	6.919	6.928	6.918	8.290	8.560

In Table A.1 and A.2 we observe that the MSE over different SNR and s is very similar for all methods. This is due to the fact that all methods find a similar support $z^\star$. Using this support all problems solve the same convex quadratic programming problem. We also observe that the reduced size $\|Z_{\texttt{BR}}\|_0 \approx 2s$ and $\|Z_{\texttt{DP}}\|_0 \approx s$. So as the problem in $(\mathcal{P}_Z)$ increases with s, MOSEK takes more time to solve $(\mathcal{P}_Z)$ and because $\|Z_{\texttt{BR}}\|_0 > \|Z_{\texttt{DP}}\|_0$ the `DP` is faster for large s.

A.3 Real Datasets

For the real datasets listed in the main paper, we present the out-sample MSE for the different methods in Table A.3.

Similar to the in-sample MSE, Table A.3 shows that `DP` delivers a lower out-sample MSE than `BR` in 4 out of 5 datasets, and `DP` also has a lower out-sample MSE than the `warm start`, `BH Alg 7` and `KDD` for all datasets. The `screening` method outperforms the `DP` on the (SC) and (CR) dataset, however as explained in the main paper for $\eta = \sqrt{N_{train}}$ the result of `screening` in Table A.3 on the (SC) and (CR) datasets is essentially the results obtained by applying the MOSEK solver to the original problem (reaching a time limit of 300 seconds).

Table A.3: Out-sample MSE on real datasets, averaged over 50 independent train-test splits. Lowest error for each dataset is highlighted in grey.

	DP $k=40$	DP $k=\hat{k}$	BR $k=40$	BR $k=\hat{k}$	warm start	screening	BH Alg 7	KDD
(FB)	3.026×10^{-4}	3.025×10^{-4}	3.022×10^{-4}	$\mathbf{3.020\times10^{-4}}$	out of memory	3.022×10^{-4}	3.203×10^{-4}	3.409×10^{-4}
(ON)	$\mathbf{1.796\times10^{-4}}$	1.797×10^{-4}	1.797×10^{-4}	1.797×10^{-4}	1.797×10^{-4}	1.797×10^{-4}	1.803×10^{-4}	1.803×10^{-4}
(SC)	1.263×10^{-2}	1.263×10^{-2}	1.398×10^{-2}	1.370×10^{-2}	1.326×10^{-2}	$\mathbf{1.257\times10^{-2}}$	1.454×10^{-2}	1.473×10^{-2}
(CR)	2.892×10^{-2}	2.891×10^{-2}	2.893×10^{-2}	2.894×10^{-2}	2.900×10^{-2}	$\mathbf{2.868\times10^{-2}}$	3.103×10^{-2}	3.148×10^{-2}
(UJ)	$\mathbf{2.149\times10^{-2}}$	2.324×10^{-2}	2.684×10^{-2}	2.691×10^{-2}	2.468×10^{-2}	2.291×10^{-2}	3.848×10^{-2}	3.080×10^{-2}

B Proof of Proposition 3.1

We provide the proof of Proposition 3.1, which is not included in the main paper.

Proof. Using the big-M equivalent formulation, we have

$$\begin{aligned}
\mathcal{J}_k^\star = \min_{\substack{z\in\{0,1\}^n \\ \sum z_j \le s}} \quad & \min \quad \sum_{i=1}^k \lambda_i y_i^2 + \langle c, x\rangle + \eta^{-1}\|x\|_2^2 \\
& \text{s.t.} \quad x \in \mathbb{R}^n,\ y \in \mathbb{R}^k \\
& \qquad \sqrt{\lambda_i} y_i = \sqrt{\lambda_i}\,\langle v_i, x\rangle \qquad i \in [k] \\
& \qquad |x_j| \le M z_j \qquad j \in [n] \\
& \qquad Ax \le b.
\end{aligned}$$

Fix a feasible solution for z and consider the inner minimization problem. By associating the first two constraints with the dual variables α and β, the Lagrangian function is defined as

$$\begin{aligned}\mathcal{L}(x,y,\alpha,\beta) &= \sum_{i=1}^{k}\lambda_i y_i^2 + \langle c,x\rangle + \eta^{-1}\|x\|_2^2 + \sum_{i=1}^{k}\alpha_i\sqrt{\lambda_i}\left(\langle v_i,x\rangle - y_i\right) + \beta^\top(Ax-b)\\ &= -\beta^\top b + y^\top\Lambda y - \alpha^\top\sqrt{\Lambda}y + \left\langle c + V\sqrt{\Lambda}\alpha + A^\top\beta, x\right\rangle + \eta^{-1}\|x\|_2^2,\end{aligned}$$

in which $\Lambda = \mathrm{diag}\{\lambda_1,\cdots,\lambda_k\}$. For any feasible solution z, the inner minimization problem is a convex quadratic optimization problem and we have

$$\mathcal{J}_k^\star = \min_{\substack{z\in\{0,1\}^n\\ \sum z_j\le s}}\ \max_{\substack{\alpha\in\mathbb{R}^k\\ \beta\in\mathbb{R}^m_+}} L(z,\alpha,\beta),$$

where the objective function L is defined as

$$L(z,\alpha,\beta) := -\beta^\top b + \min_{y\in\mathbb{R}^k}\ y^\top\Lambda y - \alpha^\top\sqrt{\Lambda}y + \min_{\substack{x\in\mathbb{R}^n\\ |x_j|\le Mz_j\ \forall j}}\left\langle c + V\sqrt{\Lambda}\alpha + A^\top\beta, x\right\rangle + \eta^{-1}\|x\|_2^2.$$

We will reformulate the two optimization subproblems in the definition of L. For any feasible pair $\beta\in\mathbb{R}^m_+$ and $\alpha\in\mathbb{R}^k$, the subproblem over y is an unconstrained convex quadratic optimization problem. The corresponding optimal solution for y is

$$y^\star(\alpha,\beta) = \frac{1}{2}(\sqrt{\Lambda})^{-1}\alpha.$$

Consequently, the optimal value of the y-subproblem is given by

$$\min_{y\in\mathbb{R}^k}\ y^\top\Lambda y - \alpha^\top\sqrt{\Lambda}y = -\frac{1}{4}\|\alpha\|_2^2.$$

Next, consider the x-subproblem. Let $\gamma := c + V\sqrt{\Lambda}\alpha + A^\top\beta$ and let γ_j denote the j-th element of γ. The big-M equivalent formulation for the x-subproblem admits the form

$$\min_{\substack{x\in\mathbb{R}^n\\ |x_j|\le Mz_j\ \forall j}}\ \sum_{j=1}^{n}\gamma_j x_j + \frac{x_j^2}{\eta} = \sum_{j=1}^{n}\ \min_{\substack{x\in\mathbb{R}^n\\ |x_j|\le Mz_j\ \forall j}}\ \gamma_j x_j + \frac{x_j^2}{\eta} = \sum_{j=1}^{n} -\frac{\eta}{4}\gamma_j^2 z_j,$$

where the last equality exploits the fact that the optimal solution of x_j is

$$x_j^\star(z_j) = \begin{cases} -\frac{\eta}{2}\gamma_j & \text{if } z_j = 1,\\ 0 & \text{if } z_j = 0.\end{cases}$$

We thus have

$$L(z,\alpha,\beta) = -\beta^\top b - \frac{1}{4}\sum_{i=1}^{k}\alpha_i^2 - \sum_{j=1}^{n}\frac{\eta}{4}\gamma_j^2 z_j,$$

where $\gamma = c + V\sqrt{\Lambda}\alpha + A^\top\beta$ and γ_j is the j-th element of γ. Rewriting the summations using norm and matrix multiplications completes the proof. □

C Principal Component Hierarchy for Sparsity-Penalized Quadratic Programs

The approach proposed in the main paper can be extended to solve the $\|\cdot\|_0$-penalized problem of the form

$$\begin{array}{ll}\min & \langle c, x\rangle + \langle x, Qx\rangle + \eta^{-1}\|x\|_2^2 + \theta\|x\|_0 \\ \text{s.t.} & x \in \mathbb{R}^n,\ Ax \leq b\end{array}$$

for some sparsity-inducing parameter $\theta > 0$. The corresponding approximation using k principal components of the matrix Q is

$$\begin{array}{rll}\mathcal{U}_k^\star \triangleq & \min & \langle c, x\rangle + \displaystyle\sum_{i=1}^{k} \lambda_i y_i^2 + \eta^{-1}\|x\|_2^2 + \theta\|x\|_0 \\ & \text{s.t.} & x \in \mathbb{R}^n,\ y \in \mathbb{R}^k \\ & & Ax \leq b \\ & & \sqrt{\lambda_i} y_i = \sqrt{\lambda_i}\,\langle v_i, x\rangle \quad i \in [k].\end{array} \tag{$\mathcal{W}_k$}$$

Proposition C.1 (Min-max characterization)**.** *For each $k \leq n$, the optimal value of problem ($\mathcal{W}_k$) is equal to*

$$\mathcal{U}_k^\star = \min_{z\in\{0,1\}^n} \ \max_{\substack{\alpha\in\mathbb{R}^k \\ \beta\in\mathbb{R}^m_+}} \ H(z, \alpha, \beta),$$

where the objective function H is defined as

$$H(z,\alpha,\beta) \triangleq \theta \sum_{j=1}^{n} z_j - \beta^\top b - \frac{1}{4}\|\alpha\|_2^2 - \frac{\eta}{4}(c + V\sqrt{\Lambda}\alpha + A^\top\beta)^\top \mathrm{diag}(z)(c + V\sqrt{\Lambda}\alpha + A^\top\beta). \tag{A.1}$$

Proof of Proposition C.1. The sparsity-penalized principal component approximation problem can be rewritten using the big-M formulation as

$$\begin{array}{lll}\displaystyle\min_{z\in\{0,1\}^n} & \min & \langle c, x\rangle + \displaystyle\sum_{i=1}^{k} \lambda_i y_i^2 + \eta^{-1}\|x\|_2^2 + \theta \sum_{j=1}^{n} z_j \\ & \text{s.t.} & x \in \mathbb{R}^n,\ y \in \mathbb{R}^k \\ & & \sqrt{\lambda_i} y_i = \sqrt{\lambda_i}\,\langle v_i, x\rangle \quad i \in [k] \\ & & |x_j| \leq M z_j \qquad\qquad j \in [n] \\ & & Ax \leq b.\end{array}$$

For any feasible solution z, the inner minimization problem is a convex quadratic optimization problem. By strong duality, we have the equivalent problem

$$\mathcal{U}_k^\star = \min_{z\in\{0,1\}^n} \ \max_{\substack{\alpha\in\mathbb{R}^k \\ \beta\in\mathbb{R}^m_+}} \ H(z, \alpha, \beta),$$

where the objective function H is

$$H(z,\alpha,\beta) = -\beta^\top b + \min_{y\in\mathbb{R}^k}\ y^\top\sqrt{\Lambda}y - \alpha^\top \mathrm{diag}(\sqrt{\Lambda})y + \min_{\substack{x\in\mathbb{R}^n \\ |x_j|\le Mz_j\ \forall j}} \left\langle c + V\mathrm{diag}(\sqrt{\Lambda})\alpha + A^\top\beta, x\right\rangle + \eta^{-1}\|x\|_2^2 + \theta\sum_{j=1}^n z_j.$$

Following proposition 3.1 we can calculate the optimal values for $y^\star$ and $x^\star$. Considering the x-subproblem, let $\gamma = c + V\sqrt{\Lambda}\alpha + A^\top\beta$ and γ_j be the j-th element of γ. The big-M equivalent formulation for the x-subproblem admits the form

$$\begin{aligned}\min_{\substack{x\in\mathbb{R}^n \\ |x_j|\le Mz_j\ \forall j}} \sum_{j=1}^n \gamma_j x_j + \frac{x_j^2}{\eta} + \theta z_j &= \sum_{j=1}^n \min_{\substack{x\in\mathbb{R}^n \\ |x_j|\le Mz_j\ \forall j}} \gamma_j x_j + \frac{x_j^2}{\eta} + \theta z_j \\ &= \sum_{j=1}^n (-\frac{\eta}{4}\gamma_j^2 + \theta) z_j, \end{aligned} \tag{A.2}$$

where the last equality exploits the fact that the optimal solution of x_j is

$$x_j^\star(z_j) = \begin{cases} -\frac{\eta}{2}\gamma_j & \text{if } \frac{\eta}{4}\gamma_j^2 > \theta, \\ 0 & \text{if } \frac{\eta}{4}\gamma_j^2 \le \theta. \end{cases}$$

As a consequence, we have

$$H(z,\alpha,\beta) = -\beta^\top b - \frac{1}{4}\sum_{i=1}^k \alpha_i^2 + \sum_{j=1}^n (-\frac{\eta}{4}\gamma_j^2 + \theta) z_j,$$

where $\gamma = c + V\sqrt{\Lambda}\alpha + A^\top\beta$ and γ_j is the j-th element of γ. Rewriting the summations using norm and matrix multiplications completes the proof. □

Lemma C.2 (Closed-form minimizer)**.** *Given any pair (α,β), the minimizer of the function H defined in (A.1) can be computed as*

$$\arg\min_{z\in\{0,1\}^n} H(z,\alpha,\beta) = \mathbb{I}\big\{\frac{\eta}{4}\mathrm{diag}((c+V\sqrt{\Lambda}\alpha+A^\top\beta)(c+V\sqrt{\Lambda}\alpha+A^\top\beta)^\top) > \theta\big\},$$

where $\mathbb{I}$ is the component-wise indicator function and the diag *operator here returns the vector of diagonal elements of the input matrix.*

This lemma immediately follows from (A.1).

Appendix B

Sparse inductive matrix completion using updated side information

A Proof of proposition 3.1

proof to 3.1

Proof. Given the diagonal projection matrices W_i we can rewrite the sum in (2.3) over known entries of $A, \sum_{(i,j)\in\Omega} (X_{ij} - A_{ij})^2$, as a sum over the rows of A :

$$\sum_{i=1}^{n} \|(x_i - a_i) W_i\|_2^2$$

where x_i is the i th row of X. Using $X = UB^T$, then $x_i = u_i B^T$ where u_i is the i th row of U. Moreover,

$$\|U\|_2^2 = \sum_{i=1}^{n} \|u_i\|_2^2$$

Then, Problem (2.3) becomes:

$$\min_U \frac{1}{nm} \left(\sum_{i=1}^{n} \left(\left\| \left(u_i B^T - a_i\right) W_i \right\|_2^2 + \frac{1}{\eta} \|u_i\|_2^2 \right) \right)$$

We then notice that within the sum $\sum_{i=1}^{n}$ each row of U can be optimized separately, leading to:

$$\frac{1}{nm} \left(\sum_{i=1}^{n} \min_{u_i} \left(\left\| \left(u_i B^T - a_i\right) W_i \right\|_2^2 + \frac{1}{\eta} \|u_i\|_2^2 \right) \right)$$

The inner optimization problem $\min_{u_i} \left\| \left(u_i B^T - a_i\right) W_i \right\|_2^2 + \frac{1}{\eta} \left\| u_i \right\|_2^2$ can be rewritten to a sparse quadratic programming problem

$$\begin{aligned} \sum_i \min_{u_i} \quad & u_i Q_i u_i^\top + c_i^\top u_i^\top + \frac{1}{\eta} \left\| u_i \right\|_2^2 \\ \text{s.t.} \quad & r_i(1 - z_i) = \mathbf{0}, \quad \forall i = 1 \dots p \\ & z \in \{0,1\}^p, \|z\|_0 \leq s \end{aligned} \tag{$\mathcal{P}$}$$

where

$$Q_i = B^\top W_i B \quad c_i = -2 a_i W_i B$$

□

B Comparing the method by Vreugdenhil et al. [2021] to the method by Xie and Deng [2020]

We compare both methods using the MAPE In Table B.1 we observe that the methods of Vreugdenhil et al. [2021] and Xie and Deng [2020] can achieve a similar MAPE for all instances which is lower compared to `fastImpute`. The methods by Vreugdenhil et al. [2021] and Xie and Deng [2020] have a comparable computational time. A possible explanation for this is that the process of rewritting (2.3) to ($\mathcal{P}$) takes up most of the computational time and therefore the difference

Table B.1: MAPE and computational time over different parameters averaged over 20 independent replications. Lower is better.

						Vreugdenhil et al. [2021]		Xie and Deng [2020]		fastImpute	
	n	m	p	μ	s	MAPE	time (s)	MAPE	time (s)	MAPE	time(s)
	10^3	10^3	10^2	0.95	10	0.39	0.18	0.39	0.18	3.00	1.77
n	10^4	10^3	10^2	0.95	10	0.39	2.09	0.39	2.08	2.80	5.62
	10^5	10^3	10^2	0.95	10	0.39	19.19	0.39	23.01	2.52	24.74
m	10^3	10^4	10^2	0.95	10	0.35	1.52	0.35	1.58	2.82	3.24
	10^3	10^5	10^2	0.95	10	0.35	20.46	0.35	21.24	2.79	9.27
p	10^3	10^3	5×10^2	0.95	10	0.39	1.74	0.39	1.47	3.55	2.82
	10^3	10^3	10^3	0.95	10	0.39	5.87	0.39	5.65	4.09	3.80
s	10^3	10^3	10^2	0.95	5	0.82	0.19	0.82	0.18	6.11	0.60
	10^3	10^3	10^2	0.95	30	0.19	0.22	0.19	0.23	5.80	89.48
μ	10^3	10^3	10^2	0.8	10	0.36	0.36	0.36	0.36	2.71	1.00
	10^3	10^3	10^2	0.99	10	2.91	0.12	2.91	0.12	21.28	2.06

between the two methods is minimal. We do also observe that both approaches are faster for all instances except two compared to `fastImpute`.

Bibliography

E. Acar, G. Gürdeniz, M. A. Rasmussen, D. Rago, L. O. Dragsted, and R. Bro. Coupled matrix factorization with sparse factors to identify potential biomarkers in metabolomics. In *2012 IEEE 12th International Conference on Data Mining Workshops*, pages 1–8, 2012. doi: 10.1109/ICDMW.2012.17.

R. Aguilera, R. Delgado, D. Dolz, and J. Aguero. Quadratic MPC with ℓ_0-input constraint. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 19:10888–10893, 08 2014.

A. Atamturk and A. Gomez. Safe screening rules for ℓ_0-regression from perspective relaxations. In *Proceedings of the 37th International Conference on Machine Learning*, pages 421–430, 2020.

M. R. Baye and D. F. Parker. Combining ridge and principal component regression:a money demand illustration. *Communications in Statistics - Theory and Methods*, 13(2):197–205, 1984.

A. Beck and Y. Eldar. Sparsity constrained nonlinear optimization: Optimality conditions and algorithms. *SIAM Journal on Optimization*, 23:1480–1509, 2012. doi: 10.1137/120869778.

A. Beck and N. Hallak. On the minimization over sparse symmetric sets: Projections, optimality conditions, and algorithms. *Mathematics of Operations Research*, 41, 09 2015.

D. Bertsimas and R. Cory-Wright. A scalable algorithm for sparse portfolio selection. *arXiv preprint arXiv:1811.00138*, 2020.

D. Bertsimas and M. L. Li. Fast exact matrix completion: A unified optimization framework for matrix completion, 2020a.

D. Bertsimas and M. L. Li. Interpretable matrix completion: A discrete optimization approach. *arXiv preprint arXiv:1812.06647*, 2020b.

D. Bertsimas and R. Shioda. Algorithm for cardinality-constrained quadratic optimization. *Computational Optimization and Applications*, 43:1–22, 05 2009. doi: 10.1007/s10589-007-9126-9.

D. Bertsimas and B. van Parys. Sparse high-dimensional regression: Exact scalable algorithms and phase transitions. *The Annals of Statistics*, 48:300–323, 2017.

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

D. Bertsimas, M. S. Copenhaver, and R. Mazumder. The trimmed Lasso: Sparsity and robustness. *arXiv preprint arXiv:1409.8033*, 2017.

D. Bertsimas, R. Cory-Wright, and J. Pauphilet. Solving large-scale sparse pca to certifiable (near) optimality. *arXiv preprint arXiv:2005.05195*, 2020a.

D. Bertsimas, R. Cory-Wright, and J. Pauphilet. A unified approach to mixed-integer optimization: Nonlinear formulations and scalable algorithms. *arXiv preprint arXiv:1907.02109*, 2020b.

S. Boyd, L. Xiao, and A. Mutapcic. Subgradient methods notes for ee392o, October 2003.

E. Candès and B. Recht. Exact matrix completion via convex optimization. *Commun. ACM*, 55 (6):111–119, 2012. ISSN 0001-0782. doi: 10.1145/2184319.2184343. URL https://doi.org/10.1145/2184319.2184343.

E. J. Candes and Y. Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2010. doi: 10.1109/JPROC.2009.2035722.

E. J. Candes and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080, 2010. doi: 10.1109/TIT.2010.2044061.

P. Chen and D. Suter. Recovering the missing components in a large noisy low-rank matrix: application to sfm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8): 1051–1063, 2004. doi: 10.1109/TPAMI.2004.52.

K.-Y. Chiang, C.-J. Hsieh, and I. S. Dhillon. Matrix completion with noisy side information. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/0609154fa35b3194026346c9cac2a248-Paper.pdf.

A. Dedieu, H. Hazimeh, and R. Mazumder. Learning sparse classifiers: Continuous and mixed integer optimization perspectives, 2020.

P. S. Dhillon, Y. Lu, D. Foster, and L. Ungar. New subsampling algorithms for fast least squares regression. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, page 360–368. Curran Associates Inc., 2013.

D. Dua and C. Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

R. Escalante and M. Raydan. *Alternating Projection Methods*. Society for Industrial and Applied Mathematics, 2011. ISBN 1611971934.

J. Fliege and B. F. Svaiter. Steepest descent methods for multicriteria optimization. *Mathematical Methods of Operations Research*, 51(3):479–494, 2000. doi: 10.1007/s001860000043.

L. E. Ghaoui and H. Lebret. Robust solutions to least-squares problems with uncertain data. *SIAM J. Matrix Anal. Appl.*, 18(4):1035–1064, 1997. doi: 10.1137/S0895479896298130.

A. Gomez and O. Prokopyev. A mixed-integer fractional optimization approach to best subset selection. *Optimization-online*, 2018.

G. Gordon and R. Tibshirani. Gradient desent revisited. https://www.cs.cmu.edu/~ggordon/10725-F12/scribes/10725_Lecture5.pdf, 2012.

J. Hannon, M. Bennett, and B. Smyth. Recommending twitter users to follow using content and collaborative filtering approaches. In *RecSys'10 - Proceedings of the 4th ACM Conference on Recommender Systems*, pages 199–206, 01 2010. doi: 10.1145/1864708.1864746.

F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), Dec. 2015. ISSN 2160-6455. doi: 10.1145/2827872. URL https://doi.org/10.1145/2827872.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, 2 edition, 2009.

T. Hastie, R. Tibshirani, and R. J. Tibshirani. Extended comparisons of best subset selection, forward stepwise selection, and the lasso. *arXiv preprint arXiv:1707.08692*, 2017.

H. Hazimeh, R. Mazumder, and A. Saab. Sparse regression at scale: Branch-and-bound rooted in first-order optimization. *arXiv preprint arXiv:2004.06152*, 2020.

A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67, 1970.

P. Jain, P. Netrapalli, and S. Sanghavi. Low-rank matrix completion using alternating minimization. *arXiv preprint arXiv:1212.0467*, 2012.

R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from a few entries. *IEEE Transactions on Information Theory*, 56(6):2980–2998, 2010. doi: 10.1109/TIT.2010.2046205.

R. Liu, J. Kuang, Q. Gong, and X. Hou. Principal component regression analysis with SPSS. *Computer Methods and Programs in Biomedicine*, 71(2):141 – 147, 2003.

J. Löfberg. YALMIP: A toolbox for modeling and optimization in MATLAB. In *IEEE International Conference on Robotics and Automation*, pages 284–289, 2004.

J. Lu, G. Liang, J. Sun, and J. Bi. A sparse interactive model for matrix completion with side information. *Advances in Neural Information Processing Systems*, 29, 2016.

R. Mazumder, P. Radchenko, and A. Dedieu. Subset selection with shrinkage: Sparse linear modeling when the SNR is low. *arXiv preprint arXiv:1708.03288*, 2020.

A. Miller. *Subset Selection in Regression*. CRC Press, 2002.

MOSEK ApS. *The MOSEK optimization toolbox. Version 9.2.*, 2019. URL https://docs.mosek.com/9.2/cmdtools/index.html.

B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227–234, 1995.

I. Nazarov, B. Shirokikh, M. Burkina, G. Fedonin, and M. Panov. Sparse group inductive matrix completion. *arXiv preprint arXiv:1804.10653*, 2018.

Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer, 2003.

X. Ning and G. Karypis. Sparse linear methods with side information for top-n recommendations. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, page 155–162, 2012. ISBN 9781450312707. doi: 10.1145/2365952.2365983. URL https://doi.org/10.1145/2365952.2365983.

T. Næs and H. Martens. Principal component regression in NIR analysis: Viewpoints, background details and selection of components. *Journal of Chemometrics*, 2(2):155–167, 1988.

B. Recht. A simpler approach to matrix completion. *Journal of Machine Learning Research*, 12 (104):3413–3430, 2011. URL http://jmlr.org/papers/v12/recht11a.html.

J. D. M. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. *Proceedings of the 22nd International Conference on Machine Learning*, page 713–719, 2005. doi: 10.1145/1102351.1102441.

M. T. Ribeiro, S. Singh, and C. Guestrin. "why should I trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

R. T. Rockafellar and R. J.-B. Wets. *Variational analysis*, volume 317. Springer Science & Business Media, 2009.

S. Si, K.-Y. Chiang, C.-J. Hsieh, N. Rao, and I. S. Dhillon. Goal-directed inductive matrix completion. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 1165–1174. Association for Computing Machinery, 2016.

V. Sindhwani, S. S. Bucak, J. Hu, and A. Mojsilovic. One-class matrix completion with low-density factorizations. In *2010 IEEE International Conference on Data Mining*, pages 1055–1060, 2010. doi: 10.1109/ICDM.2010.164.

A. Soni, T. Chevalier, and S. Jain. Noisy inductive matrix completion under sparse factor models. *arXiv preprint arXiv:1609.03958*, 2016.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B (Methodological)*, 58(1):267–288, 1996.

S. Tu, R. Boczar, M. Simchowitz, M. Soltanolkotabi, and B. Recht. Low-rank solutions of linear matrix equations via procrustes flow, 2016.

M. Udell and A. Townsend. Why are big data matrices approximately low rank? *SIAM Journal on Mathematics of Data Science*, 1(1):144–160, 2019.

J. Vig, S. Sen, and J. Riedl. The tag genome: Encoding community knowledge to support novel interaction. *ACM Trans. Interact. Intell. Syst.*, 2(3), 2012. ISSN 2160-6455. doi: 10.1145/2362394.2362395.

R. Vreugdenhil, V. A. Nguyen, A. Eftekhari, and P. M. Esfahani. Principal component hierarchy for sparse quadratic programs. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10607–10616. PMLR, 2021.

Z. Weng and X. Wang. Low-rank matrix completion for array signal processing. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2697–2700, 2012. doi: 10.1109/ICASSP.2012.6288473.

W. Xie and X. Deng. Scalable algorithms for the sparse ridge regression. *arXiv preprint arXiv:1806.03756*, 2020.

M. Xu, R. Jin, and Z.-H. Zhou. Speedup matrix completion with side information: Application to multi-label learning. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper/2013/file/e58cc5ca94270acaceed13bc82dfedf7-Paper.pdf.

G. Yuan, L. Shen, and W.-S. Zheng. A block decomposition algorithm for sparse optimization. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 275–285, 2020.

X. Zhang, S. Du, and Q. Gu. Fast and sample efficient inductive matrix completion via multi-phase procrustes flow. *Proceedings of the 35th International Conference on Machine Learning*, 80:5756–5765, 2018.

T. Zhao, Z. Wang, and H. Liu. A nonconvex optimization framework for low rank matrix estimation. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 559–567. MIT Press, 2015.

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B (Methodological)*, 67:301–320, 2005.