APPROXIMATION ALGORITHMS FOR MIXED INTEGER NON-LINEAR OPTIMIZATION PROBLEMS

A Dissertation
Presented to
The Academic Faculty

By

Guanyi Wang

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
Algorithms, Combinatorics and Optimization (ACO)
H. Milton Stewart School of Industrial and Systems Engineering

Georgia Institute of Technology

May 2021

APPROXIMATION ALGORITHMS FOR MIXED INTEGER NON-LINEAR OPTIMIZATION PROBLEMS

Thesis committee:

Dr. Santanu S. Dey, Advisor
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Yao Xie
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Mohit Singh
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Santosh Vempala
College of Computing
Georgia Institute of Technology

Dr. Rahul Mazumder
Sloan School of Management
Massachusetts Institute of Technology

Dr. Marco Molinaro, Reader
Computer Science Department
Pontifical Catholic University of Rio de Janeiro

Date approved: March 30, 2021

To my parents

ACKNOWLEDGMENTS

First, I would like to express my deepest gratitude to my advisor Santanu S. Dey for his guidance and support throughout my Ph.D., for his kindness, patience, enthusiasm, and immense knowledge. I am fortunate to have many research discussions and walkings with him around campus, during which he shared some exciting research ideas. It was indeed a pleasure to work with him. I would also like to extend my sincere gratitude to my committee: Santosh Vempala, Yao Xie, Rahul Mazumder, Mohit Singh, and Marco Molinaro, for taking the time to attend my defense and give insightful comments on my thesis. I also wish to thank all my collaborators throughout my Ph.D. at Georgia Tech, Santanu S. Dey, Rahul Mazumder,Marco Molinaro, and Yao Xie.

I was lucky to have a significant cohort at Georgia Tech. I am glad I went through the PhD with Minshuo Chen, Zhehui Chen, Haoming Jiang, Tianyi Liu, Yan Li, Yuyang Shi, Liyan Xie, Weijun Xie, Wanrong Zhang. Special thanks to the students with related research interests, Digvijay Boob, Xiaoyi Gu, Asteroide Santana, Uthaipon (Tao) Tantipongpipat, Yu Yang for the helpful discussions on both ACO course study and research. I would also like to acknowledge the assistance from Tianyi Chen, Fangzheng Xie, Xilei Zhao for their assistance on the job search.

Lastly, I would like to thank my parents for always supporting, encouraging, and providing me with many excellent educational opportunities.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF NOTATIONS

Notations for Chapter 1

M	number of samples
d	number of features in each sample
$\boldsymbol{x}_m$	the m-th sample
$\boldsymbol{X}$	data matrix
$\boldsymbol{A}$	sample covariance matrix
$\boldsymbol{v}$	leading principal component
PCA	principal component analysis
SPCA	sparse principal component analysis
k	sparsity parameter
$\|\cdot\|_2, \|\cdot\|_1, \|\cdot\|_0$	ℓ_2, ℓ_1, ℓ_0 norm
$\lambda^k(\boldsymbol{A})$	optimal value of the SPCA problem
conv(S)	convex hull of set S
$[d]$	short notation of index set $\{1, \ldots, d\}$
diag$(\boldsymbol{v})$	diagonal matrix generated from a given vector $\boldsymbol{v}$
Tr$(\boldsymbol{A})$	trace of a matrix $\boldsymbol{A}$

opt_{ℓ_1}	optimal value of the ℓ_1-relaxation of SPCA
ρ	multiplicative approximation ratio such that $\text{opt}_{\ell_1}/\lambda^k(\boldsymbol{A}) \leq \rho^2$
$\{\lambda_i, \boldsymbol{w}_i\}_{i=1}^d$	eigenpair of sample covariance matrix $\boldsymbol{A}$
λ_{TH}	thresholding parameter for eigenvalues
I^+, I^-	$I^+ := \{i : \lambda_i > \lambda\},\ \ I^- := \{i : \lambda_i \leq \lambda\}$
$\{g_i\}_{i=1}^d$	continuous variable $g_i := \boldsymbol{v}^\top \boldsymbol{w}_i$
$\{\theta_i\}_{i=1}^d$	upper bound of g_i
N	number of splitting points in one branch
$\{\gamma_i^j\}_{j=-N}^N$	splitting points of interval $[-\theta_i, \theta_i]$ for each i
$\{\xi_i\}_{i=1}^n$	piecewise linear upper approximation of g_i^2
s	upper bound of $\sum_{i \in I^-} -(\lambda_i - \lambda) g_i^2$
$\bar{\lambda}$	$\bar{\lambda} := \max\{\lambda_i : \lambda_i \leq \lambda\}$
$\{\lambda_{i_j}\}_{j=1}^p$	$\lambda_{i_1} \geq \cdots \geq \lambda_{i_p} \geq 0$ distinct values of eigenvalues of $\boldsymbol{A}$
$\Delta\lambda$	eigenvalue gap $\Delta\lambda = \min\{\lambda_{i_j} - \lambda_{i_{j+1}}\}$ for $j = 1, \ldots, p-1$
$\bar{\boldsymbol{A}}$	perturbed covariance matrix of $\boldsymbol{A}$
$(\bar{\boldsymbol{v}}, \bar{g}, \bar{\xi}, \bar{\eta}, \bar{s})$	optimal solution for Convex-IP
$\text{opt}_{\text{convex-IP}}$	optimal value of Convex-IP
$\text{opt}_{\text{pert-convex-IP}}$	optimal value of Pert-Convex-IP
$b_{(\boldsymbol{c})}$	parameter for cutting plane

S_k	feasible region of SPCA with sparsity parameter k
T_k	feasible region of ℓ_1-relaxation with sparsity parameter k
I_{pos}	the size of set I^+
$I_{\text{pos}}^{\text{ini}}$	input size of I^+
iter	number of iterations used for Pert-Convex-IP

Notations for Chapter 2

Here we ignore the notations that have been stated in Chapter 1 with the same definitions.

r	number of principal components for row-sparse PCA problem
$\boldsymbol{V}$	top-r k-sparse principal components
rsPCA	row-sparse principal component analysis
$\|\cdot\|_{\text{op}}, \|\cdot\|_0$	operator norm, row-sparsity norm
$\mathcal{F}$	feasible region of rsPCA
$\text{opt}^{\mathcal{F}}$	optimal value of rsPCA
$\mathcal{CR}1$	first convex relaxation of $\mathcal{F}$
$\mathcal{CR}2$	second convex relaxation of $\mathcal{F}$
$\text{ub}^{\mathcal{CR}i}$	optimal value of CIP using $\mathcal{CR}i$ relaxation
S	support set with size k
$\rho_{\mathcal{CR}i}$	multiplicative approximation ratio between $\mathcal{CR}i$ and $\mathcal{F}$
Δ_j^{out}	value for leaving candidate of j-th row

Δ_j^{in}	value for entering candidate of j-th row

Notations for Chapter 3

d	number of features in each input sample
n	number of samples
$\boldsymbol{X}_i$	i-th input sample
Y_i	i-th output sample
$\boldsymbol{X}$	input sample matrix
$\boldsymbol{Y}$	output sample matrix
$\boldsymbol{\beta}$	parameter with respect to input sample
β_0	bias parameter
ReLU-reg	Regression problem with ReLU activation function
z^{OPT}	optimal value of ReLU-reg
$(\boldsymbol{\beta}^{\text{opt}}, \beta_0^{\text{opt}})$	optimal solution of the ReLU-regression problem
$(\boldsymbol{\beta}^*, \beta_0^*)$	true parameter in an underlying statistical model
ϵ	noise in an underlying statistical model
$\mathcal{I}^+$	index set with positive outputs, to be concise, assume $\mathcal{I}^+ := \{i \in [n] : Y_i > 0\} = \{1, \ldots, m\}$
$\mathcal{I}^-$	index set with non-positive outputs, to be concise, assume $\mathcal{I}^- := \{i \in [n] : Y_i \leq 0\} = \{m+1, \ldots, n\}$, thus $\mathcal{I}^+ \cup \mathcal{I}^- = [n]$

$\phi(\cdot, \cdot)$	loss function with respect to non-positive outputs
I	active set such that $I \subseteq [m] = \mathcal{I}^+$
I^{opt}	active set $I^{\text{opt}} \subseteq [m] = \mathcal{I}^+$ with respect to objective solution of ReLU-reg
$P(I)$	set of feasible region of $(\boldsymbol{\beta}, \beta_0)$
$f_I(\boldsymbol{\beta}, \beta_0)$	objective function corresponding to $P(I)$
inner-phase	optimization problem $\min_{(\boldsymbol{\beta}, \beta_0) \in P(I)} f_I(\boldsymbol{\beta}, \beta_0) + \phi(\boldsymbol{\beta}, \beta_0)$
$z^*(I)$	optimal value of inner-phase optimization problem given I
$\sigma(\cdot, \cdot)$	auxiliary convex upper approximation function $\sigma : \mathbb{R} \times \mathbb{R}_+ \mapsto \mathbb{R}$
$f_I^\sigma(\boldsymbol{\beta}, \beta_0)$	objective function corresponding to $P(I)$ using σ
$z^\sigma(I)$	optimal value of inner-phase convex upper approximation given I
$(\boldsymbol{\beta}^I, \beta_0^I)$	optimal solution of inner-phase convex upper approximation given I
z^{approx}	objective value obtained from the generalized approximation algorithm
$(\hat{\boldsymbol{\beta}}, \hat{\beta}_0)$	solution obtained from the generalized approximation algorithm
PE	prediction error
Obj	objective value
RE	recovery error
GE	generalization error

SUMMARY

Mixed integer non-linear optimization (MINLO) problems are usually NP-hard. Although obtaining feasible solutions is relatively easy via heuristic or local search methods, it is still challenging to guarantee the quality (i.e., the gap to optimal value) of a given feasible solution even under mild assumptions in a tractable fashion. In this thesis, we propose efficient mixed integer linear programming based algorithms for finding feasible solutions and proving the quality of these solutions for three widely-applied MINLO problems.

In Chapter 1, we study the sparse principal component analysis (SPCA) problem. SPCA is a dimensionality reduction tool in statistics. Comparing with the classical principal component analysis (PCA), the SPCA enhances the interpretability by incorporating an additional sparsity constraint in the feature weights (factor loadings). However, unlike PCA, solving the SPCA problem to optimality is NP-hard. Most conventional methods for SPCA are heuristics with no guarantees such as certificates of optimality on the solution-quality via associated *dual bounds*. We present a convex integer programming (IP) framework to derive dual bounds based on the ℓ_1-relaxation of SPCA. We show the theoretical worst-case guarantee of the dual bounds provided by the convex IP. Based on numerical results, we empirically illustrate that our convex IP framework outperforms existing SPCA methods in both accuracy and efficiency of finding dual bounds. Moreover, these dual bounds obtained in computations are significantly better than worst-case theoretical guarantees.

Chapter 2 focuses on solving a non-trivial generalization of SPCA – the (row) sparse principal component analysis (rsPCA) problem. Solving rsPCA is to find the top-r leading principal components of a covariance matrix such that all these principal components share the same support set with cardinality at most k. In this chapter, we propose: (a) a convex integer programming relaxation of rsPCA that gives upper (dual) bounds for rsPCA, and; (b) a new local search algorithm for finding primal feasible solutions for rsPCA. We also show that, in the worst-case, the dual bounds provided by the convex IP are within an

affine function of the optimal value. We demonstrate our techniques applied to large-scale covariance matrices.

In Chapter 3, we consider a fundamental training problem of finding the best-fitting ReLU concerning square-loss – also called "ReLU Regression" in machine learning. We begin by proving the NP-hardness of the ReLU regression. We then present an approximation algorithm to solve the ReLU regression, whose running time is $\mathcal{O}(n^k)$ where n is the number of samples, and k is a predefined integral constant as an algorithm parameter. We analyze the performance of this algorithm under two regimes and show that: (1) given an arbitrary set of training samples, the algorithm guarantees an (n/k)-approximation for the ReLU regression problem – to the best of our knowledge, this is the first time that an algorithm guarantees an approximation ratio for arbitrary data scenario; thus, in the ideal case (i.e., when the training error is zero) the approximation algorithm achieves the globally optimal solution for the ReLU regression problem; and (2) given training sample with Gaussian noise, the same approximation algorithm achieves a much better asymptotic approximation ratio which is independent of the number of samples n. Extensive numerical studies show that our approximation algorithm can perform better than the classical gradient descent algorithm in ReLU regression. Moreover, numerical results also imply that the proposed approximation algorithm could provide a good initialization for gradient descent and significantly improve the performance.

CHAPTER 1

USING ℓ_1-RELAXATION AND INTEGER PROGRAMMING TO OBTAIN DUAL BOUNDS FOR SPARSE PCA

This chapter is based on a joint work with Santanu S. Dey, Rahul Mazumder, and Guanyi Wang, [1].

1.1 Introduction

Principal component analysis (PCA) is one of the most widely used dimensionality reduction methods in data science. Given a data matrix

$$\boldsymbol{X} = \begin{pmatrix} - & \boldsymbol{x}_1^\top & - \\ & \vdots & \\ - & \boldsymbol{x}_M^\top & - \end{pmatrix} \in \mathbb{R}^{M \times d}$$

with M samples and d features in each sample; and each feature is centered to have zero mean, PCA seeks to find a principal component direction $\boldsymbol{v} \in \mathbb{R}^d$ with $\|\boldsymbol{v}\|_2 = 1$ that maximizes the variance of a weighted combination of features. Formally, this PC direction can be found by solving

$$\max_{\|\boldsymbol{v}\|_2=1} \boldsymbol{v}^\top \boldsymbol{A} \boldsymbol{v} \tag{PCA}$$

where $\boldsymbol{A} := \frac{1}{M}\boldsymbol{X}^\top \boldsymbol{X} = \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{x}_m \boldsymbol{x}_m^\top$ is the sample covariance matrix. An obvious drawback of PCA is that all the entries of $\hat{\boldsymbol{v}}$ (an optimal solution to PCA) are (usually) nonzero, which leads to the PC direction being a linear combination of all features – this impedes interpretability [2, 3, 4]. In biomedical applications for example, when $\boldsymbol{X}$ corresponds to the gene-expression measurements for different samples, it is desirable to obtain

a PC direction which involves only a handful of the features (e.g, genes) for interpretation purposes. In financial applications (e.g, $\boldsymbol{A}$ may denote a covariance matrix of stock-returns), a sparse subset of stocks that are responsible for driving the first PC direction may be desirable for interpretation purposes. Indeed, in many scientific and industrial applications [5, 6, 7], for additional interpretability, it is desirable for the factor loadings to be sparse, i.e., few of the entries in $\hat{\boldsymbol{x}}$ are nonzero and the rest are zero. This motivates the notion of a sparse principal component analysis (SPCA) [6, 3], wherein, in addition to maximizing the variance, one also desires the direction of the first PC to be sparse in the factor loadings. The most natural optimization formulation of this problem, modifies criterion with an additional sparsity constraint on $\boldsymbol{x}$ leading to:

$$\lambda^k(\boldsymbol{A}) := \max_{\|\boldsymbol{v}\|_2=1, \|\boldsymbol{v}\|_0 \leq k} \boldsymbol{v}^\top \boldsymbol{A}\boldsymbol{v} \tag{SPCA}$$

where $\|\boldsymbol{v}\|_0 \leq k$ is equivalent to allowing at most k components of x to be nonzero.

Many heuristic algorithms for SPCA have been proposed in the literature that use greedy methods [3, 8, 9, 10], alternating methods [11] and the related power methods [12]. However, conditions under which (some of) these computationally friendlier methods can be shown to work well, make very strong and often unverifiable assumptions on the problem data. Therefore, the performance of these heuristics (in terms of how close they are to an optimal solution of the SPCA problem) on a given dataset is not clear.

Unlike the PCA problem, the SPCA problem is known to be NP-hard [13, 14]. Chan et al. [13] study the computational complexity of solving Sparse PCA approximately. The authors (1) present an efficient algorithm to achieve an $d^{-1/3}$-approximation; (2) show that SPCA is NP-hard to approximate within $(1-\epsilon)$ for some constant $\epsilon > 0$; (3) show Small-Set Expansion (SSE) hardness for any polynomial-time constant factor approximation algorithm, (4) give a "quasi-quasi-polynomial" gap for the standard SDP relaxation. Chowdhury et al. [15] present three polynomial-time approximation algorithms for SPCA

which provides "sparse" solutions (may not satisfies the sparsity constraint) with provable bounds. In contrast, Papailiopoulos et al. [16] introduce a combinatorial algorithm for SPCA by examining a finite set of vectors in a low-dimensional eigen-subspace of $\boldsymbol{A}$. This combinatorial algorithm returns a primal feasible solution of SPCA in time $O(d^{\text{rank}(\boldsymbol{A})})$ with provable approximation guarantees that depend on the eigenvalues of $\boldsymbol{A}$.

Since SPCA is NP-hard, there has been exciting work in the statistics community [17, 18] in understanding the statistical properties of convex relaxations (e.g., those proposed by [19] and variants) of SPCA. It has been established [17, 18] that the statistical performance of estimators available from convex relaxations are sub-optimal (under suitable modeling assumptions) when compared to estimators obtained by (optimally) solving SPCA—this further underlines the importance of creating tools to be able to solve SPCA to optimality.

Our main goal in this paper is to propose an integer programming framework that allows the computation of certificates of optimality via dual bounds, which make limited restrictive/unverifiable assumptions on the data. Dual bounds also translate to suitable guarantees in statistical performance of the estimator—see for example, [20][Theorem 4] for results pertaining to approximate solutions for sparse regression settings[1]. To the best of our knowledge, the only published methods for obtaining dual bounds of SPCA are based on semidefinite programming (SDP) relaxations [21, 22, 10, 23] (see Appendix A.1 for the SDP relaxation) and spectral methods involving a low-rank approximation of the matrix $\boldsymbol{A}$ [16]. Both these approaches however, have some limitations. The SDP relaxation does not appear to scale easily (using off-the-shelf solver Mosek 8.0.0.60) for matrices with more than a few hundred rows/columns, while applications can be significantly larger. Indeed, even a relatively recent implementation based on the Alternating Direction Method of Multipliers for solving the SDP considers instances with size $d \approx 200$ [24]. The spectral methods involving a low-rank approximation of $\boldsymbol{A}$ proposed in [16] have a running time

[1] In [20], estimators with certificates on dual bounds translate to simple modifications of error bounds that correspond to the global solution of the original nonconvex estimator.

of $\mathcal{O}(d^r)$ where r is the rank of the matrix—in order to scale to large instances, no more than a rank 2 approximation of the original matrix seems possible. The paper [25] presents a specialized branch and bound solver[2] to obtain solutions to the SPCA problem, but their method can handle problems with $d \approx 100$ – the approach presented here is different, and our proposal scales to problem instances that are much larger.

The methods proposed here are able to obtain approximate dual bounds of SPCA by solving convex integer programs and a related perturbed version of convex integer programs that are easier to solve. The dual bounds we obtain are incomparable to dual bounds based on the SDP relaxation, i.e. neither dominates the other, and the method appears to scale well to matrices up to sizes of 2000×2000.

1.2 Main results

In this paper, we use bold upper case letters such as $\boldsymbol{A}, \boldsymbol{X}$ to denote symmetric matrices. The (i, j)-th component of matrix $\boldsymbol{A}$ is denoted as $[\boldsymbol{A}]_{ij}$ or $\boldsymbol{A}_{ij}$ in short. We use bold lower case letters such as $\boldsymbol{v}, \boldsymbol{x}$ for vectors, and denote the i-th component of a vector $\boldsymbol{v}$ as $[\boldsymbol{v}]_i$ or $\boldsymbol{v}_i$ in short. We use upper case letter I for set of indices. Given a vector where $\boldsymbol{v} \in \mathbb{R}^n$ and $I \subseteq [d]$, we let $\boldsymbol{v}_I \in \mathbb{R}^d$ to be the vector:

$$[\boldsymbol{v}_I]_i := \begin{cases} \boldsymbol{v}_i & i \in I \\ 0 & i \notin I \end{cases}$$

We use the usual notation $\|\cdot\|_1,\ \|\cdot\|_2$ for $\ell_1,\ \ell_2$ norm respectively for a given vector. Let $\|\cdot\|_0$ be the ℓ_0 norm which denotes the number of non-zero components. Given a set S, we denote conv(S) as the convex hull of S; given a positive integer d we denote $\{1, \ldots, d\}$ by $[d]$; given a matrix $\boldsymbol{A}$, we denote its trace by tr$(\boldsymbol{A})$. Given d scalars $\boldsymbol{v}_1, \ldots \boldsymbol{v}_d$, diag$(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d)$ is the $d \times d$ matrix whose diagonal elements are $\boldsymbol{v}_i$'s and the off-diagonal terms are equal to 0.

[2]This paper is not available in the public domain at the time of writing this paper.

In SPCA, the constraint $\|\boldsymbol{v}\|_2 = 1, \|\boldsymbol{v}\|_0 \leq k$ implies that $\|\boldsymbol{v}\|_1 \leq \sqrt{k}$. Thus, one obtains the so-called ℓ_1-norm relaxation of SPCA:

$$\text{opt}_{\ell_1} := \max_{\|\boldsymbol{v}\|_2 \leq 1, \|\boldsymbol{v}\|_1 \leq \sqrt{k}} \boldsymbol{v}^\top \boldsymbol{A} \boldsymbol{v}. \qquad (\ell_1\text{-relax})$$

The relaxation ℓ_1-relax has two advantages:

(a) As shown in 1 below, ℓ_1-relax gives a constant factor bound on SPCA,

(b) The feasible region is convex and all the nonconvexity is in the objective function.

We build on these two advantages: our convex IP relaxation is a further relaxation of ℓ_1-relax (together with some implied linear inequalities for SPCA) which heavily use the fact that the feasible region of ℓ_1-relax is convex. We require to use IP methods and construct the convex IP, since the objective of ℓ_1-relax is non-convex. Thus, we use a combination of ℓ_1-relax and IP methods to obtain strong dual bounds.

We note that ℓ_1-relax is an important estimator in its own right [7, 6] – it is commonly used in the statistics/machine-learning community as one that leads to an eigenvector of $\boldsymbol{A}$ with entries having a small ℓ_1-norm (as opposed to a small ℓ_0-norm). We emphasize that ℓ_1-relaxation has never been used to computationally obtain dual bounds for SPCA. Indeed, to the best of our knowledge there has been no systematic study of the theoretical and empirical computational properties of the ℓ_1-relaxation vis-à-vis SPCA.

The rest of this section is organized as follows: In 1.2.1, we present the constant factor bound on SPCA given by ℓ_1-relax, improving upon some known results. In Section 1.2.2, we present the construction of our convex IP and prove results on the quality of bound provided. In Section 1.2.3, we discuss perturbing the original matrix in order to make the convex IP more efficiently solvable while still providing reasonable dual bounds. In Section 1.4, we present results from our computational experiments.

1.2.1 Quality of ℓ_1-relaxation as a surrogate for the SPCA problem

The following theorem is an improved version of a result appearing in [26] (Exercise 10.3.7).

Theorem 1. *The objective value opt_{ℓ_1} is upper bounded by a multiplicative factor ρ^2 away from $\lambda^k(\boldsymbol{A})$, i.e., $\lambda^k(\boldsymbol{A}) \leq opt_{\ell_1} \leq \rho^2 \cdot \lambda^k(\boldsymbol{A})$ with $\rho \leq 1 + \sqrt{\frac{k}{k+1}}$.*

Proof of Theorem 1 is provided in Section 1.3. While we have improved upon the bound presented in [26], we do not know if this new bound is tight. The approximation ratio $1 + \sqrt{\frac{k}{k+1}}$ from Theorem 1 yields an almost 100% gap (see formal definition of gap in Section 4) in the worst case. From a practitioners' viewpoint, a 100% gap is obviously far from ideal and would not be considered as "solving" the problem. However, as we shall see in Section 4, the ℓ_1-relaxation does provide very good dual bounds in many instances. Moreover, as stated above the approximation ratio of $1 + \sqrt{\frac{k}{k+1}}$ is the best we can prove; however this bound may be significantly away from the actual bound.

Theorem 1 has implications regarding existence of polynomial-time algorithms to obtain a constant-factor approximation guarantee for ℓ_1-relax. In particular, the proof of Theorem 1 implies that if one can obtain a solution for ℓ_1-relax which is within a constant factor, say θ, of opt_{ℓ_1}, then a solution for SPCA problem can be obtained, which is within a constant factor (at most $\theta\rho^2 \approx 4\theta$) of $\lambda^k(\boldsymbol{A})$. Therefore, the ℓ_1-relaxation is also inapproximable in general.

1.2.2 From ℓ_1-relaxation to convex integer programming model

A classical integer programming approach to finding dual bounds of SPCA would be to go to an extended space involving the product of $\boldsymbol{v}$-variables and include one binary variable per $\boldsymbol{v}$-variable in order to model the ℓ_0-norm constraint, resulting in a very large number of

binary variables. In particular, a typical model could be of the form:

$$\begin{array}{ll} \max & \text{tr}(\boldsymbol{A}\boldsymbol{V}) \\ \text{s.t.} & \sum_{j=1}^{n} \boldsymbol{z}_i \leq k, \qquad \boldsymbol{z} \in \{0,1\}^d \\ & \|\boldsymbol{v}\|_2 \leq 1, \qquad \boldsymbol{v}_i \in [-\boldsymbol{z}_i, \boldsymbol{z}_i], \ \ \forall\, i = 1, \ldots, d\ . \\ & \begin{pmatrix} 1 & \boldsymbol{v}^\top \\ \boldsymbol{v} & \boldsymbol{V} \end{pmatrix} \succeq \boldsymbol{0}, \quad \text{rank} \begin{pmatrix} 1 & \boldsymbol{v}^\top \\ \boldsymbol{v} & \boldsymbol{V} \end{pmatrix} = 1 \end{array}$$

It is easy to see that such a model is challenging due to (a) d binary variables (b) "quadratic" increase in number of variables ($\boldsymbol{V}$) and (c) the presence of the rank constraint. Even with significant progress, it is well-known that solving such problems beyond d being a few hundred variables is extremely challenging [27, 28]. Indeed, instances with an arbitrary quadratic objective and bound constraints cannot be generally solved (exactly) by modern state-of-the-art methods as soon as the number of variables exceed a hundred or so [29, 30, 31, 32, 33].

This is how we address the challenges discussed above.

1. d binary variables (a): the feasible region of ℓ_1-relax is a convex set. Therefore, we do not have to include binary variables to model the ℓ_0-norm constraint. We will use ℓ_1-relax as our basic relaxation.

2. Quadratic increase in number of variables (b) and rank constraint (c): We do not use the $\boldsymbol{V}$ variables to model the quadratic objective. Instead we upper bound the quadratic objective using piecewise linear function via integer programming techniques.

In other words, since the feasible region of ℓ_1-relax is a convex set and takes care of challenge (a), we model/upper bound the objective function using IP techniques to deal with challenges (b) and (c). Specifically, we follow the following procedure:

step-0: By spectral decomposition, let $\boldsymbol{A} = \sum_{i=1}^{d} \lambda_i \boldsymbol{w}_i \boldsymbol{w}_i^\top$ where $(\lambda_i)_{i=1}^d, (\boldsymbol{w}_i)_{i=1}^d$ are unit

norm orthogonal eigen-pairs. Then the objective function of ℓ_1-relax is:

$$\sum_{i=1}^{d} \lambda_i (\boldsymbol{v}^\top \boldsymbol{w}_i)^2.$$

step-1: Given any thresholding parameter λ_{TH}, the eigenvalues of $\boldsymbol{A}$ can be split into two sets based on that thresholding parameter λ_{TH},

$$I^+ := \{i : \lambda_i > \lambda_{\mathrm{TH}}\} \quad \text{and} \quad I^- := \{i : \lambda_i \leq \lambda_{\mathrm{TH}}\}.$$

Thus the objective function can be represented as a sum of λ_{TH} and two parts that depend on the index sets defined above,

$$\lambda_{\mathrm{TH}} + \sum_{i \in I^+} (\lambda_i - \lambda_{\mathrm{TH}})(\boldsymbol{v}^\top \boldsymbol{w}_i)^2 + \sum_{i \in I^-} (\lambda_i - \lambda_{\mathrm{TH}})(\boldsymbol{v}^\top \boldsymbol{w}_i)^2.$$

Note that the first term is convex and the second term is concave. Since the objective is a maximizing, we need to deal with the first term. This idea of splitting the objective function into convex and concave part is a well-studied approach for attacking non-convex quadratic objective functions. See for example [34, 35] for use of some similar ideas.

step-2: For each index $i \in I^+$, replace $\boldsymbol{v}^\top \boldsymbol{w}_i$ with a single continuous variable g_i, and set

$$\theta_i := \max\{\boldsymbol{v}^\top \boldsymbol{w}_i : \|\boldsymbol{v}\|_2 \leq 1, \|\boldsymbol{v}\|_0 \leq k\}.$$

Then for each g_i with $i \in I^+$, construct a piecewise linear upper approximation ξ_i for

g_i^2 with $g_i \in [-\theta_i, \theta_i]$ using the following piecewise linear approximation (PLA) set,

$$\mathrm{PLA} := \left\{ (g,\xi,\eta) : \begin{array}{l} g_{ji} = \boldsymbol{a}_j^\top \boldsymbol{v}_i,\ (j,i) \in [d]\times[r], \\ g_{ji} = \sum_{\ell=-N}^{N} \gamma_{ji}^{\ell} \eta_{ji}^{\ell} \\ \xi_{ji} = \sum_{\ell=-N}^{N} \left(\gamma_{ji}^{\ell}\right)^2 \eta_{ji}^{\ell} \\ \left(\eta_{ji}^{\ell}\right)_{\ell=-N}^{N} \in \text{SOS-II} \end{array} \right\},$$

where the (SOS-II) denotes the *special ordered sets of type 2* constraints [36] as follows: for any i and j,

$$\text{SOS-II} := \left\{ \left(\eta_{ji}^{\ell}\right)_{\ell=-N}^{N} : \begin{array}{lll} \sum_{\ell=-N}^{N-1} z_{ji}^{\ell} = 1 & & \\ z_{ji}^{\ell} \in \{0,1\} & \forall\, \ell = -N,\ldots,N-1 \\ \eta_{ji}^{\ell} + \eta_{ji}^{\ell+1} \le z_{ji}^{\ell} & \forall\, \ell = -N,\ldots,N-1 \\ \eta_{ji}^{\ell} \ge 0 & \forall\, \ell = -N,\ldots,N \end{array} \right\}.$$

step-3: For index set I^-, since $\lambda_i - \lambda_{\mathrm{TH}} < 0$ for all $i \in I^-$, we obtain a convex constraint

$$\sum_{i\in I^-} -(\lambda_i - \lambda_{\mathrm{TH}})(\boldsymbol{v}^\top \boldsymbol{w}_i)^2 \le s$$

Therefore, a convex integer programming problem is obtained as follows:

$$\begin{array}{ll} \max & \lambda_{\mathrm{TH}} + \sum_{i\in I^+}(\lambda_i - \lambda_{\mathrm{TH}})\xi_i - s =: \mathrm{opt}_{\text{convex-IP}} \\ \text{s.t.} & \|\boldsymbol{v}\|_2 \le 1, \quad \|\boldsymbol{v}\|_1 \le \sqrt{k} \\ & g_i = \boldsymbol{v}^\top \boldsymbol{w}_i, \quad g_i \in [-\theta_i, \theta_i], \quad \forall\, i = 1,\ldots,d \\ & \left\{ \begin{array}{l} g_i = \sum_{j=-N}^{N} \gamma_i^j \eta_i^j, \quad \xi_i = \sum_{j=-N}^{N} (\gamma_i^j)^2 \eta_i^j \\ (\eta_i^{-N},\ldots,\eta_i^N) \in \text{SOS-2}, \quad i \in I^+ \end{array} \right. \qquad \text{(Convex-IP)} \\ & \sum_{i\in I^+} \left(\xi_i - \frac{\theta_i^2}{4N^2}\right) + \sum_{i\in I^-} g_i^2 \le 1 \\ & \sum_{i\in I^-} -(\lambda_i - \lambda_{\mathrm{TH}}) g_i^2 \le s \end{array}$$

Notations and explanations of Convex-IP model:

ℓ_1 constraints: The first row of constraints $\|\boldsymbol{v}\|_2 \leq 1, \ \ \|\boldsymbol{v}\|_1 \leq \sqrt{k}$.

Variable g_i: The second row of constraints $g_i = \boldsymbol{v}^\top \boldsymbol{w}_i, \quad g_i \in [-\theta_i, \theta_i], \quad \forall\, i = 1, \ldots, d$ transfers the product terms $\boldsymbol{v}^\top \boldsymbol{w}_i$ into a single variable for each $i \in [d]$.

Variable ξ_i: The third bracket of constraints

$$\begin{cases} g_i = \sum_{j=-N}^{N} \gamma_i^j \eta_i^j, \quad \xi_i = \sum_{j=-N}^{N} (\gamma_i^j)^2 \eta_i^j \\ (\eta_i^{-N}, \ldots, \eta_i^N) \in \text{SOS-2}, \quad i \in I^+ \end{cases}$$

forms ξ_i as a piecewise-linear upper approximation of g_i^2 based on a classic integer programming technique–SOS-2. Let $2N+1$ be the number of splitting points of the domain $[-\theta_i, \theta_i]$ of variable g_i, where the set of splitting points $(\gamma_i^j)_{j=-N}^N$ satisfy

$$-\theta_i = \gamma_i^{-N} < \ldots \gamma_i^0 \ (= 0) < \ldots < \gamma_i^N = \theta_i.$$

Without any prior information of the optimal solution, we partition the set $[-\theta_i, \theta_i]$ equally to minimize the (worst-case) upper bounds, i.e., by letting $(\gamma_i^j)_{j=-N}^N \leftarrow \left(\frac{j}{N} \cdot \theta_i\right)_{j=-N}^N$ be the value of j^{th} splitting point. See Section A.3 for details.

Quadratic constraints: The forth row of constraints does the following: Since $\boldsymbol{w}_i$'s are orthonormal, then $\|\boldsymbol{v}\|_2 \leq 1$ implies $\|g_i\| \leq 1$ for all $i = 1, \ldots, d$. Together with ξ_i representing g_i^2, we can obtain the implied inequality:

$$\sum_{i \in I^+} \xi_i + \sum_{i \in I^-} g_i^2 \leq 1 + \sum_{i \in I^+} \frac{\theta_i^2}{4N^2}$$

The second term in the right-hand-side reflects the fact that ξ_i is not exactly equal to g_i^2, but only a piecewise linear upper bound of g_i^2. Note that the exact value of the second term in the right-hand-side also depends on the way one splits the set

$[-\theta_i, \theta_i]$, the value $\sum_{i\in I^+} \frac{\theta_i^2}{4N^2}$ in above formula is obtained via splitting $[-\theta_i, \theta_i]$ equally, which can be shown as the minimum upper bounds without any prior idea of the optimal solution $\boldsymbol{v}$ of SPCA or ℓ_1-relax. See the proof in Section A.3 for details. This constraint (cutting-plane) is not necessarily needed for a correct model – it is used since it helps improving the dual bound of the LP relaxation and significantly improves the running-time of the solver.

Convex constraint: The final constraint

$$\sum_{i\in I^-} -(\lambda_i - \lambda_{\text{TH}}) g_i^2 \leq s \qquad \text{(convex-constraint)}$$

is a convex constraint which can be obtained in step-3 where $\boldsymbol{v}^\top \boldsymbol{w}_i$ is replaced by a variable g_i.

Therefore, we arrive at the following result:

Proposition 1.2.1. *The optimal objective value $opt_{\text{convex-IP}}$ of Convex-IP is an upper bound on the SPCA problem.*

Proposition 1.2.1 is formally verified in Appendix A.2.

Next combining the result of Theorem 1 with the quality of the approximation of the objective function of ℓ_1-relax by Convex-IP, we obtain the following result:

Proposition 1.2.2. *The optimal objective value $opt_{\text{convex-IP}}$ of Convex-IP is upper bounded by*

$$OPT_{\text{convex-IP}} \leq \rho^2 \lambda^k(\boldsymbol{A}) + \frac{1}{4N^2} \sum_{i\in I^+} (\lambda_i - \lambda_{\text{TH}}) \theta_i^2.$$

A proof of Proposition 1.2.2 is presented in Appendix A.3.

Finally, let us discuss why we expect Convex-IP to be appealing from a computational viewpoint. Unlike typical integer programming approaches, the number of binary variables

in Convex-IP is $(2N+1)\cdot|I^+|$ which is usually significantly smaller than d. Indeed, heuristics for SPCA generally produce good values of λ, and in almost all experiments we found that $|I^+| \ll n$ due to the choice of thresholding parameter λ_{TH}. Moreover, N is a parameter we control. In order to highlight the "computational tractability" of Convex-IP, we formally state the following result:

Proposition 1.2.3. *Assuming the number of splitting points N and the size of set I^+ is fixed, the Convex-IP problem can be solved in polynomial time.*

Note that the convex integer programming method which is solvable in polynomial time, does not contradict the inapproxamability of the SPCA problem, since $\mathrm{opt}_{\text{convex-IP}}$ is upper bounded by the sum of $\rho^2\lambda^k(\boldsymbol{A})$ and a term corresponding to the sample covariance matrix.

1.2.3 Improving the running time of Convex-IP

Perturbation of the covariance matrix $\boldsymbol{A}$:

In practice, we do the following (sequence of) perturbation on covariance matrix $\boldsymbol{A}$ to reduce the running time of solving convex IP. Again let λ (obtained from some heuristic method) be a lower bound on the $\lambda^k(\boldsymbol{A})$, let $\boldsymbol{A} = \sum_{i=1}^{d}\lambda_i\boldsymbol{w}_i\boldsymbol{w}_i^\top$ be the spectral decomposition of $\boldsymbol{A}$ with $\lambda_1 \geq \ldots \geq \lambda_d \geq 0$.

1. Set parameter $\bar{\lambda} := \max\{\lambda_i : \lambda_i \leq \lambda_{\mathrm{TH}}\}$. To be concise, we assume $\bar{\lambda} < \lambda_{\mathrm{TH}}$. However, when $\bar{\lambda} = \lambda_{\mathrm{TH}} = \max\{\lambda_i : \lambda_i \leq \lambda_{\mathrm{TH}}\}$, one can apply Algorithm 1 to obtain a matrix $\bar{\boldsymbol{A}} \succeq \boldsymbol{A}$ such that none of the eigenvalues of $\bar{\boldsymbol{A}}$ equals λ_{TH}. We then replace $\boldsymbol{A}$ by $\bar{\boldsymbol{A}}$. Let $\bar{\lambda}_1, \ldots, \bar{\lambda}_n$ be the eigenvalues of (the updated) $\bar{\boldsymbol{A}}$ and let $\bar{\lambda} := \max\{\bar{\lambda}_i : \bar{\lambda}_i \leq \lambda_{\mathrm{TH}}\}$, we obtain that $\bar{\lambda} < \lambda_{\mathrm{TH}}$ for $\bar{\boldsymbol{A}}$.

2. Perturb the covariance matrix $\boldsymbol{A} = \sum_{i=1}^{n}\lambda_i\boldsymbol{w}_i\boldsymbol{w}_i^\top$ to get $\bar{\boldsymbol{A}} = \sum_{i\in I^+}\lambda_i\boldsymbol{w}_i\boldsymbol{w}_i^\top + \sum_{i\in I^-}\bar{\lambda}\boldsymbol{w}_i\boldsymbol{w}_i^\top$. Note that the objective value $\mathrm{opt}_{\text{convex-IP}}(\bar{\boldsymbol{A}})$ in Convex-IP is an upper

Algorithm 1 Perturbation of $\boldsymbol{A}$

Input: Sample covariance matrix $\boldsymbol{A}$ and threshold λ_{TH}.

Output: A perturbed sample covariance matrix $\bar{\boldsymbol{A}}$ with distinct eigenvalues such that $\bar{\boldsymbol{A}} \succeq \boldsymbol{A}$ and *none* of the eigenvalues of $\bar{\boldsymbol{A}}$ equals λ.

1: Compute eigenvalue decomposition of $\boldsymbol{A}$ as $\boldsymbol{A} = \boldsymbol{W}^\top \boldsymbol{\Lambda} \boldsymbol{W}$ with $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, and sort all distinct distinct eigenvalues in $\boldsymbol{\Lambda}$ as

$$\lambda_{i_1} > \cdots > \lambda_{\mathrm{TH}} = \lambda_{i_j} > \cdots \lambda_{i_p} \geq 0, \quad \text{where } p \leq n.$$

2: Set $\Delta\lambda \leftarrow \min\{\lambda_{i_j} - \lambda_{i_{j+1}} \,|\, j = 1, \ldots, p-1\}$ as the minimum eigen-gap.

3: Set perturbed diagonal $\bar{\boldsymbol{\Lambda}} \leftarrow \boldsymbol{\Lambda} + \mathrm{diag}\left(\frac{i-1}{n}\epsilon \,|\, i = n, \ldots, 1\right)$ with $\epsilon = \frac{1}{2}\Delta\lambda$.

4: **return** $\bar{\boldsymbol{A}} \leftarrow \boldsymbol{V}^\top \bar{\boldsymbol{\Lambda}} \boldsymbol{V}$.

bound on $\mathrm{opt}_{\text{convex-IP}}(\boldsymbol{A})$. This is because if $(\boldsymbol{v}, g, \xi, \eta, s)$ is a feasible solution of Convex-IP model, then the objective function value of Convex-IP corresponding to $\bar{\boldsymbol{A}}$ is at least as large as that of $\boldsymbol{A}$. Replace $\boldsymbol{A}$ by $\bar{\boldsymbol{A}}$.

3. Therefore, the convex constraint $\sum_{i \in I^-} -(\lambda_i - \lambda_{\mathrm{TH}}) g_i^2 \leq s$ in Convex-IP can be replaced by $\sum_{i \in I^-} -(\bar{\lambda} - \lambda_{\mathrm{TH}}) g_i^2 \leq s$, i.e., $\sum_{i \in I^-} g_i^2 \leq \frac{s}{\lambda_{\mathrm{TH}} - \bar{\lambda}}$.

4. Let $(\bar{\boldsymbol{x}}, \bar{g}, \bar{\xi}, \bar{\eta}, \bar{s})$ be an optimal solution for Convex-IP. Since the convex constraint achieves equality for any optimal solution of Convex-IP, i.e., (a) $\sum_{i \in I^-} -(\lambda_{\mathrm{TH}} - \bar{\lambda}) \bar{g}_i^2 = \bar{s}$ together with (b) $\sum_{i=1}^n \bar{g}_i^2 = \sum_{i \in I^-} \bar{g}_i^2 + \sum_{i \in I^+} \bar{g}_i^2 \leq 1$ and (c) $1 \leq \sum_{i \in I^+} \bar{\xi}_i + \sum_{i \in I^-} \bar{g}_i^2 \leq 1 + \frac{1}{4N^2} \sum_{i \in I^+} \theta_i^2$ imply the following inequalities:

$$1 - \frac{\bar{s}}{\lambda_{\mathrm{TH}} - \bar{\lambda}} \leq \sum_{i \in I^+} \bar{\xi}_i \leq 1 + \frac{1}{4N^2} \sum_{i \in I^+} \theta_i^2 - \frac{\bar{s}}{\lambda_{\mathrm{TH}} - \bar{\lambda}}, \quad \sum_{i \in I^+} \bar{g}_i^2 \leq 1 - \frac{\bar{s}}{\lambda_{\mathrm{TH}} - \bar{\lambda}}.$$

Thus a simplified convex IP corresponding to the perturbed covariance matrix is:

$$
\begin{aligned}
\max \quad & \lambda_{\mathrm{TH}} + \textstyle\sum_{i\in I^+}(\lambda_i - \lambda_{\mathrm{TH}})\xi_i - s =: \mathrm{opt}_{\text{pert-convex-IP}} \\
\text{s.t.} \quad & \|\boldsymbol{v}\|_2 \le 1, \quad \|\boldsymbol{v}\|_1 \le \sqrt{k} \\
& g_i = \boldsymbol{v}^\top \boldsymbol{w}_i, \quad g_i \in [-\theta_i, \theta_i], \quad i \in I^+ \\
& \begin{cases} g_i = \sum_{j=-N}^{N} \gamma_i^j \eta_i^j, \quad \xi_i = \sum_{j=-N}^{N} (\gamma_i^j)^2 \eta_i^j \\ (\eta_i^{-N}, \ldots, \eta_i^N) \in \text{SOS-2}, \quad i \in I^+ \end{cases} \\
& 1 - \tfrac{s}{\lambda_{\mathrm{TH}} - \bar{\lambda}} \le \textstyle\sum_{i\in I^+} \xi_i \le 1 + \sum_{i\in I^+} \tfrac{\theta_i^2}{4N^2} - \tfrac{s}{\lambda_{\mathrm{TH}} - \bar{\lambda}} \\
& \textstyle\sum_{i\in I^+} g_i^2 \le 1 - \tfrac{s}{\lambda_{\mathrm{TH}} - \bar{\lambda}} \\
& \boldsymbol{c}^\top |\boldsymbol{v}| \le b_{(\boldsymbol{c})}
\end{aligned}
\tag{Pert-Convex-IP}
$$

where the quadratic constraints in Pert-Convex-IP are updated based on the discussion above and the final constraint $\boldsymbol{c}^\top |\boldsymbol{v}| \le b_{(\boldsymbol{c})}$ represents the cutting planes that we add, see Proposition 1.2.5 for details.

Proposition 1.2.4. *The optimal objective value $opt_{\text{Pert-Convex-IP}}$ is upper bounded by*

$$
OPT_{\text{Pert-Convex-IP}} \le \rho^2 \lambda^k(\boldsymbol{A}) + \rho^2(\bar{\lambda} - \lambda_{\min}(\boldsymbol{A})) + \frac{1}{4N^2} \sum_{i\in I^+} (\lambda_i - \lambda_{\mathrm{TH}})\theta_i^2.
$$

Note that in Pert-Convex-IP, we do not need the variables $g_i, i \in I^-$ which greatly reduces the number of variables since in general $|I^+| \ll n$. In practice, we note a significant reduction in running time, while the dual bound obtained from Pert-Convex-IP model remains reasonable. More details are presented in Section 1.4.

Refining the splitting points

Since the Pert-Convex-IP model runs much faster than the Convex-IP model, we run the Pert-Convex-IP model iteratively. In each new iteration, we add one extra splitting point describing each ξ_i function. In particular, once we solve the Pert-Convex-IP model, we add one splitting point at the optimal value of g_i.

Cutting planes

Proposition 1.2.5. *Let $\boldsymbol{v} \in \mathbb{R}^d$ be any feasible solution of SPCA. Let $|\boldsymbol{v}_{i_1}| \geq |\boldsymbol{v}_{i_2}| \geq \cdots \geq |\boldsymbol{v}_{i_{d-1}}| \geq |\boldsymbol{v}_{i_d}|$. Then let $\mathbf{c}$ be the cut:*

$$\boldsymbol{c}_{i_j} = \begin{cases} |\boldsymbol{v}_{i_j}| & \text{if } j \leq k \\ |\boldsymbol{v}_{i_k}| & \text{if } j > k. \end{cases} \tag{1.1}$$

Also let $b_{(\boldsymbol{c})} := \|(\boldsymbol{c}_{i_1}, \boldsymbol{c}_{i_2}, \boldsymbol{c}_{i_3}, \ldots, \boldsymbol{c}_{i_k})\|_2$. The inequality

$$\boldsymbol{c}^\top \boldsymbol{v} \leq b_{(\boldsymbol{c})}, \tag{1.2}$$

is a valid inequality for SPCA.

The validity of this inequality is clear: If $\boldsymbol{v}$ is a feasible point of SPCA, then the support of $\boldsymbol{v}$ is at most k and $\|\boldsymbol{v}\|_2 \leq 1$. Therefore, $\boldsymbol{c}^\top \boldsymbol{v} \leq \|(\boldsymbol{c}_{i_1}, \boldsymbol{c}_{i_2}, \boldsymbol{c}_{i_3}, \ldots, \boldsymbol{c}_{i_k})\|_2 = b_{(\boldsymbol{c})}$. Notice that this inequality is not valid for ℓ_1-relax. Also see [37]. We add these inequalities at the end of each iteration for the model where the seeding x for constructing v is chosen to be the optimal solution of the previous iteration.

1.3 Proof of Theorem 1

Given a vector $\boldsymbol{v} \in \mathbb{R}^d$, we denote the j^{th} coordinate of $\boldsymbol{v}$ as $\boldsymbol{v}_j$, and for some $J \subseteq [d]$ we denote the projection of $\boldsymbol{v}$ onto the coordinates in the index set J as $\boldsymbol{v}_J$. Define

$$S_k := \{\boldsymbol{v} \in \mathbb{R}^d \,|\, \|\boldsymbol{v}\|_2 \leq 1, \|\boldsymbol{v}\|_0 \leq k\}, \tag{1.3}$$

$$T_k := \{\boldsymbol{v} \in \mathbb{R}^d \,|\, \|\boldsymbol{v}\|_2 \leq 1, \|\boldsymbol{v}\|_1 \leq \sqrt{k}\}. \tag{1.4}$$

Note that any $\boldsymbol{v} \in T_k$ can be represented as a nonnegative combination of points in S_k, i.e., $\boldsymbol{v} = \boldsymbol{v}^1 + \cdots + \boldsymbol{v}^{\lceil d/k \rceil}$ and $\boldsymbol{v}^i \in S_k$ for all i. Here we think of each $\boldsymbol{v}^i$ as a projection onto

some unique k components of $\boldsymbol{v}$ and setting the other components to $zero$. Let $\boldsymbol{y}^i = \frac{\boldsymbol{v}^i}{\|\boldsymbol{v}^i\|_2}$, then $\boldsymbol{y}^i \in S_k$. Now we have, $\boldsymbol{v} = \sum_{i=1}^{\lceil d/k \rceil} \|\boldsymbol{v}^i\|_2 \cdot \boldsymbol{y}^i$, and therefore

$$\frac{1}{\sum_{i=1}^{\lceil d/k \rceil} \|\boldsymbol{v}^i\|_2} \boldsymbol{v} = \sum_{i=1}^{\lceil d/k \rceil} \frac{\|\boldsymbol{v}^i\|_2}{\sum_{i=1}^{\lceil d/k \rceil} \|\boldsymbol{v}^i\|_2} \cdot \boldsymbol{y}^i. \tag{1.5}$$

Thus, if we scale $\boldsymbol{v} \in T_k$ by $\|\boldsymbol{v}^1\|_2 + \ldots + \|\boldsymbol{v}^{\lceil d/k \rceil}\|_2$, then the resulting vector belongs to conv(S_k). Since we want this scaling factor to be as small as possible, we solve the following optimization problem:

$$\min \|\boldsymbol{v}^1\|_2 + \ldots + \|\boldsymbol{v}^{\lceil d/k \rceil}\|_2 : \ \boldsymbol{v} = \boldsymbol{v}^1 + \ldots + \boldsymbol{v}^{\lceil d/k \rceil};\ \boldsymbol{v}^i \in S_k,\ i \in [\lceil d/k \rceil]. \quad \text{(Bound)}$$

Without loss of generality, we assume that $\boldsymbol{v} \geq 0$ and $\boldsymbol{v}_1 \geq \boldsymbol{v}_2 \geq \cdots \geq \boldsymbol{v}_d \geq 0$. Let $\boldsymbol{v} = \bar{\boldsymbol{v}}^1 + \ldots + \bar{\boldsymbol{v}}^{\lceil d/k \rceil}$ where $\boldsymbol{v}^1, \ldots, \boldsymbol{v}^{\lceil d/k \rceil} \in S_k$ is an optimal solution of (Bound). The following proposition presents a result on an optimal solution of (Bound).

Proposition 1.3.1. *Let $I^1, \ldots, I^{\lceil d/k \rceil}$ be a collection of supports such that: I^1 indexes the k largest (in absolute value) components in x, I^2 indexes the second k largest (in absolute value) components in x, and so on. Then $I^1, \ldots, I^{\lceil d/k \rceil}$ is an optimal set of supports for* (Bound).

Proof. We prove this result by the method of contradiction. Suppose we have an optimal representation as $\boldsymbol{v} = \bar{\boldsymbol{v}}^1 + \cdots + \bar{\boldsymbol{v}}^{\lceil d/k \rceil}$ — and without loss of generality, we assume that $\|\bar{\boldsymbol{v}}^1\|_2 \geq \cdots \geq \|\bar{\boldsymbol{v}}^{\lceil d/k \rceil}\|_2$. Let $\bar{I}^1, \ldots, \bar{I}^{\lceil d/k \rceil}$ be the set of supports of $\bar{\boldsymbol{v}}^1, \ldots, \bar{\boldsymbol{v}}^{\lceil d/k \rceil}$ respectively, where we assume that the indices within each support vector are ordered such that

$$(\boldsymbol{v}_{\bar{I}^j})_1 \geq (\boldsymbol{v}_{\bar{I}^j})_2 \geq \cdots \geq (\boldsymbol{v}_{\bar{I}^j})_g$$

for all $j \in \{1, \ldots, \lceil d/k \rceil\}$ (note that $g = k$ if $j < \lceil d/k \rceil$).

Let $\bar{I}^p$ be the first support that is different from I^p, i.e., $\bar{I}^1 = I^1, \ldots, \bar{I}^{p-1} = I^{p-1}$ and $\bar{I}^p \neq I^p$. Let I^p_q be the first index in I^p that does not belong to $\bar{I}^p$ with $q \leq k$ since

$\|\bar{I}^p\|_0 = k$. Therefore, I^p_q must be in $\bar{I}^{p'}$ where $p' > p$. Note now that by construction of I and our assumption on $\bar{I}$, we have that $(\boldsymbol{v}_{I^p})_q \geq (\boldsymbol{v}_{\bar{I}^p})_q \geq (\boldsymbol{v}_{\bar{I}^p})_k$. Now we exchange the index I^p_q in $\bar{I}^{p'}$ with $\bar{I}^p_k$ in $\bar{I}^p$. We have:

$$\sqrt{\|\boldsymbol{v}_{\bar{I}^p}\|_2^2 + ((\boldsymbol{v}_{I^p})_q)^2 - ((\boldsymbol{v}_{\bar{I}^p})_k)^2} + \sqrt{\|\boldsymbol{v}_{\bar{I}^{p'}}\|_2^2 + ((\boldsymbol{v}_{\bar{I}^p})_k)^2 - ((\boldsymbol{v}_{I^p})_q)^2} \leq \|\boldsymbol{v}_{\bar{I}^p}\|_2 + \|\boldsymbol{v}_{\bar{I}^{p'}}\|_2, \tag{1.6}$$

which holds because $\|\boldsymbol{v}_{\bar{I}^p}\|_2 \geq \|\boldsymbol{v}_{\bar{I}^{p'}}\|_2$ and $((\boldsymbol{v}_{I^p})_q)^2 - ((\boldsymbol{v}_{\bar{I}^p})_k)^2 \geq 0$.

Now repeating the above step, we obtain the result. □

Based on Proposition 1.3.1, for any fixed $\boldsymbol{v} \in T_k$, we can find out an optimal solution of (Bound) in closed form. Now we would like to know, for which vector $\boldsymbol{v}$, the scaling factor $\|\boldsymbol{v}^1\|_2 + \ldots + \|\boldsymbol{v}^{\lceil d/k \rceil}\|_2$ will be the maximized. Let ρ be obtained by solving the following optimization problem:

$$\begin{aligned}
\rho = \max_{\boldsymbol{v}} \quad & \|\boldsymbol{v}_{I^1}\|_2 + \cdots + \|\boldsymbol{v}_{I^{\lceil d/k \rceil}}\|_2 \\
\text{s.t.} \quad & \boldsymbol{v} = \boldsymbol{v}_{I^1} + \cdots + \boldsymbol{v}_{I^{\lceil d/k \rceil}} \\
& \|\boldsymbol{v}\|_2^2 = \|\boldsymbol{v}_{I^1}\|_2^2 + \cdots + \|\boldsymbol{v}_{I^{\lceil d/k \rceil}}\|_2^2 \leq 1 \qquad \text{(Approximation ratio)} \\
& \|\boldsymbol{v}\|_1 = \|\boldsymbol{v}_{I^1}\|_1 + \cdots + \|\boldsymbol{v}_{I^{\lceil d/k \rceil}}\|_1 \leq \sqrt{k} \\
& \boldsymbol{v}_1 \geq \cdots \geq \boldsymbol{v}_n \geq 0.
\end{aligned}$$

Then we obtain

$$T_k \subseteq \rho \cdot \text{Conv}\left(S_k\right). \tag{1.7}$$

Although the optimal objective value of Approximation ratio is hard to compute exactly, we can still find an upper bound.

Lemma 1.3.1. *The objective value ρ of Approximation ratio is bounded from above by $1 + \sqrt{\frac{k}{k+1}}$.*

Proof. First consider the case when $d \leq 2k$. In this case, $\lceil d/k \rceil \leq 2$. Consider the optimization problem:

$$\begin{aligned} \theta = \max \quad & u + v \\ \text{s.t.} \quad & u^2 + v^2 \leq 1 \end{aligned}$$

If we think of $\|\boldsymbol{v}_{I^1}\|_2$ as u and $\|\boldsymbol{v}_{I^2}\|_2$ as v, then we see that the above problem is a relaxation of Approximation ratio and therefore $\theta = \sqrt{2}$ is an upper bound on ρ. Noting that $\sqrt{2} \leq 1 + \sqrt{\frac{k}{k+1}}$ for all $k \geq 1$, we have the result.

Now we assume that $d > 2k$ and consequently $\lceil d/k \rceil > 2$. From Approximation ratio, let $\|\boldsymbol{v}_{I^1}\|_1 = t$ and $\|\boldsymbol{v}_{I^1}\|_2 = \gamma$. Based on the standard relationship between ℓ_1 and ℓ_2 norm, we have

$$\gamma \leq t \leq \sqrt{k}\gamma.$$

Since each coordinate of $\boldsymbol{v}_{I^2}$ is smaller in magnitude than the average coordinate of $\boldsymbol{v}_{I^1}$, we have

$$\|\boldsymbol{v}_{I^2}\|_2 \leq \sqrt{\left(\frac{\|\boldsymbol{v}_{I^2}\|_1}{k}\right)^2 k} = \frac{t}{\sqrt{k}}. \tag{1.8}$$

Also note that an alternative bound is given by

$$\|\boldsymbol{v}_{I^2}\|_2 \leq \sqrt{1 - \gamma^2}.$$

Using an argument similar to the one used to obtain (1.8), we obtain that

$$\sum_{i=3}^{\lceil d/k \rceil} \|\boldsymbol{v}_{I^i}\|_2 \leq \sum_{i=2}^{\lceil d/k \rceil - 1} \sqrt{\left(\frac{\|\boldsymbol{v}_{I^i}\|_1}{k}\right)^2 k} = \frac{1}{\sqrt{k}} \sum_{i=2}^{\lceil d/k \rceil - 1} \|\boldsymbol{v}_{I^i}\|_1 \leq \frac{\sqrt{k} - t}{\sqrt{k}}.$$

Therefore we obtain

$$\sum_{i=1}^{\lceil d/k \rceil} \|\boldsymbol{v}_{I^i}\|_2 = \|\boldsymbol{v}_{I^1}\|_2 + \|\boldsymbol{v}_{I^2}\|_2 + \sum_{i=3}^{\lceil d/k \rceil} \|\boldsymbol{v}_{I^i}\|_2 \leq \gamma + \min\left\{\frac{t}{\sqrt{k}}, \sqrt{1-\gamma^2}\right\} + 1 - \frac{t}{\sqrt{k}}. \tag{Upper-Bound}$$

Now we consider two cases:

1. If $\frac{t}{\sqrt{k}} \geq \sqrt{1-\gamma^2}$, then Upper-Bound becomes $\gamma + \sqrt{1-\gamma^2} + 1 - \frac{t}{\sqrt{k}}$. Since $\gamma \geq \frac{t}{\sqrt{k}} \geq \sqrt{1-\gamma^2}$, γ satisfies $\gamma \geq \frac{1}{\sqrt{2}}$. Moreover we have that $t \geq \gamma, t \geq \sqrt{k(1-\gamma^2)}$. Since $\gamma \leq \sqrt{k(1-\gamma^2)}$ iff $\gamma \leq \sqrt{\frac{k}{k+1}}$ we obtain two cases:

$$\begin{aligned} \gamma + \sqrt{1-\gamma^2} + 1 - \frac{t}{\sqrt{k}} &\leq \begin{cases} \gamma + \sqrt{1-\gamma^2} + 1 - \sqrt{1-\gamma^2} & \text{if } \gamma \in \left[\frac{1}{\sqrt{2}}, \sqrt{\frac{k}{k+1}}\right] \\ \gamma + \sqrt{1-\gamma^2} + 1 - \frac{\gamma}{\sqrt{k}} & \text{if } \gamma \in \left[\sqrt{\frac{k}{k+1}}, 1\right] \end{cases} \\ &\leq \begin{cases} 1 + \sqrt{\frac{k}{k+1}} \\ 1 + \sqrt{\frac{k}{k+1}} \end{cases} \end{aligned} \tag{1.9}$$

   where (i) the first inequality holds when $\gamma = \sqrt{\frac{k}{k+1}}$, (ii) the second inequality holds since the function $f(\gamma) = \gamma + \sqrt{1-\gamma^2} + 1 - \frac{\gamma}{\sqrt{k}}$ achieves (local and global) maximum at point $\gamma = \sqrt{\frac{k+1-2\sqrt{k}}{2k+1-2\sqrt{k}}}$ which is less than $\sqrt{\frac{k}{k+1}}$ for $k = 1, 2, \ldots$, thus $f(\gamma) \leq \max\left\{f\left(\sqrt{\frac{k}{k+1}}\right), f(1)\right\} = 1 + \sqrt{\frac{k}{k+1}}$ for part $\gamma \in \left[\sqrt{\frac{k}{k+1}}, 1\right]$.

2. If $\frac{t}{\sqrt{k}} \leq \sqrt{1-\gamma^2}$, then Upper-Bound becomes $\gamma + 1$. Note now that $\frac{\gamma}{\sqrt{k}} \leq \frac{t}{\sqrt{k}} \leq \sqrt{1-\gamma^2}$, implies that γ satisfies $\gamma \leq \sqrt{\frac{k}{k+1}}$. Therefore, $1 + \gamma \leq 1 + \sqrt{\frac{k}{k+1}}$.

Therefore, this upper bound holds. □

Now we show Theorem 1 holds.

Proof. Proof of Theorem 1. Since $T_k \subseteq \rho \cdot \text{Conv}\,(S_k)$ with $\rho \leq 1 + \sqrt{\frac{k}{k+1}}$ and the objective function is maximizing a convex function, we obtain that $\lambda^k(\boldsymbol{A}) \leq \text{opt}_{\ell_1} \leq \rho^2 \cdot \lambda^k(\boldsymbol{A})$. □

1.4 Numerical experiments

In this section, we report results on our empirical comparison of the performances of (Convex-IP) method, (Pert-Convex-IP) method and the (SDP) relaxation method.

1.4.1 Hardware and Software

All numerical experiments are implemented on MacBookPro13 with 2 GHz Intel Core i5 CPU and 8 GB 1867 MHz LPDDR3 Memory. Convex-IPs were solved using Gurobi 7.0.2. SDPs were solved using Mosek 8.0.0.60.

1.4.2 Obtaining primal solutions

We used a heuristic, which is very similar to the truncated power method [11], but has some advantages over the truncated power method. Given $\boldsymbol{v} \in \mathbb{R}^d$, let $I_k(\boldsymbol{v})$ be the set of indices corresponding to the top k entries of $\boldsymbol{v}$ (in absolute value).

We start with a random initialization $\boldsymbol{v}^0$ such that $\|\boldsymbol{v}^0\|_2 = 1$, and set $I^0 \leftarrow I_k(\boldsymbol{A}^{1/2}\boldsymbol{v}^0)$ where $\boldsymbol{A}^{1/2}$ is the square root of $\boldsymbol{A}$. In the i^{th} iteration, we update

$$I^i \leftarrow I_k(\boldsymbol{A}^{1/2}\boldsymbol{v}^i),\ \boldsymbol{v}^{i+1} \leftarrow \underset{\|\boldsymbol{v}\|_2=1}{\arg\max}\ \boldsymbol{v}^\top \boldsymbol{A}_{I^i}\boldsymbol{v} \tag{1.10}$$

where $\boldsymbol{A}_I \in \mathbb{R}^{d\times d}$ is the matrix with $[\boldsymbol{A}_I]_{i,j} = [\boldsymbol{A}]_{i,j}$ for all $i, j \in I$ and $[\boldsymbol{A}_I]_{i,j} = 0$ otherwise. It is easy to see that $\boldsymbol{v}^1, \boldsymbol{v}^2, \ldots$ satisfy the constraint $\|\boldsymbol{v}\|_0 \le k$. Moreover, since $\boldsymbol{A}$ is a PSD matrix, $(\boldsymbol{v}^{i+1})^\top \boldsymbol{A}\boldsymbol{v}^{i+1} \ge (\boldsymbol{v}^i)^\top \boldsymbol{A}\boldsymbol{v}^i$ for all i. Therefore, in each iteration, the above heuristic method leads to an improved feasible solution for the (SPCA) problem.

Our method has two clear advantages over the truncated power method:

- We use standard and efficient numerical linear algebra methods to compute eigenvalues of small $k \times k$ matrices.
- The termination criteria used in our algorithm is also simple: if $I^i = I^{i'}$ for some

$i' < i$, then we stop. Clearly, this leads to a finite termination criteria.

In practice, we stop using a stopping criterion based on improvement and number of iterations instead of checking $I^i = I^{i'}$. Details are presented in Algorithm 2.

Algorithm 2 Primal Algorithm

Input: Sample covariance matrix A, cardinality constraint k, initial vector $\boldsymbol{v}^0$.
Output: A feasible solution $\hat{\boldsymbol{v}}$ of SPCA, and its objective value.

1: Start with an initial (randomized) vector $\boldsymbol{v}^0$ such that $\|\boldsymbol{v}^0\|_2 = 1$ and $\|\boldsymbol{v}^0\|_0 \leq k$.
2: Set the initial current objective value Obj $\leftarrow (\boldsymbol{v}^0)^\top \boldsymbol{A}\boldsymbol{v}^0$.
3: Set the initial past objective value $\tilde{\text{obj}} \leftarrow 0$.
4: Set the maximum number of iterations be $i^{\max}$.
5: **while** obj $-$ $\tilde{\text{obj}} > \epsilon$ and $i \leq i^{\max}$ **do**
6: Set $\tilde{\text{obj}} \leftarrow$ obj.
7: Set $I^i \leftarrow I_k(\boldsymbol{A}^{1/2}\boldsymbol{v}^i)$.
8: Set $x^{i+1} \leftarrow \arg\max_{\|\boldsymbol{v}\|_2=1} \boldsymbol{v}^\top \boldsymbol{A}_{I^i}\boldsymbol{v}$.
9: Set obj $\leftarrow (\boldsymbol{v}^{i+1})^\top \boldsymbol{A}\boldsymbol{v}^{i+1}$.
10: **end while**
11: **return** $\hat{\boldsymbol{v}}$ as the final $\boldsymbol{v}$ obtained from while-loop, and obj.

We use the values of $\epsilon = 10^{-6}$ and $i^{\max} = 20$ in our experiments in Algorithm 2. We repeat this algorithm with multiple random initializations. We repeat 20 times and take the best solution. We emphasize that Algorithm 2 may not lead to a global solution of (SPCA).

Our Algorithm may also be interpreted as a version of the "alternating method" used regularly as a heuristic for bilinear programs as the sparse PCA problem can be equivalently rewritten as $\max\{\boldsymbol{v}^\top \boldsymbol{A}\boldsymbol{u} : \|\boldsymbol{v}\|_2 = \|\boldsymbol{u}\|_2 = 1, \|\boldsymbol{v}\|_0 \leq k, \|\boldsymbol{u}\|_0 \leq k\}$. We have compared our primal method to two standard heuristics for finding primal feasible solutions of the sparse PCA problems in the literature: truncated power method (TPM, [38]), generalized power method (GPM, [12]) with ℓ_0-penalty. The performances of all these methods are quite similar to our method (in terms of primal solutions) on the real instances; see details in Appendix A.8.

1.4.3 Implementation of Convex-IP model and Pert-Convex-IP model

Deciding $\lambda_{\rm TH}$, N

1. Deciding λ: The size of the set $\{i : \lambda_i > \lambda_{\rm TH}\}$ denoted by $I_{\rm pos}$ plays an important role for the computational tractability of our method. So our algorithm inputs an initial value, $I_{\rm pos}^{\rm ini}$. From the primal heuristic, we obtain a lower bound $\text{LB}^{\rm primal}$ on $\lambda^k(\boldsymbol{A})$. Let $\lambda_{i_1} \geq \lambda_{i_2} \geq \cdots \geq \lambda_{i_n}$ be the eigenvalues of $\boldsymbol{A}$. If $\lambda_{i_{I_{\rm pos}^{\rm ini}}} < \text{LB}^{\rm primal}$, then we set $\lambda_{\rm TH} := \lambda_{i_{I_{\rm pos}^{\rm ini}}}$. On the other hand, if $\lambda_{i_{I_{\rm pos}^{\rm ini}}} > \text{LB}^{\rm primal}$, then let l be the smallest index such that $\lambda_{i_l} > \text{LB}^{\rm primal}$ and we set $\lambda_{\rm TH} := \lambda_{i_l}$.

2. Deciding N: In practice, θ_i was found to be significantly smaller than 1. So we used a value of $N = 3$ in all our experiments.

Final details

A total time of 7200 seconds were given to each instance for running the convex IP (any extra time reported in the tables is due to running time of singular value decomposition and primal heuristics). We have run all our experiments with $k = 10, 20$. For the (Convex-IP) method, we use: $(I_{\rm pos}^{\rm ini}, N) = (10, 3)$. For the (Pert-Convex-IP) method, let "iter" be the maximum number of iterations. We used three settings in our experiments:

$$(I_{\rm pos}^{\rm ini}, N, \text{iter}) \in \{(5, 3, 10),\ (10, 3, 3),\ (15, 3, 2)\}\,.$$

The overall algorithms using the Pert-Convex-IP model and the Convex-IP model are presented in Appendix A.6.

1.4.4 Data Sets

We conduct numerical experiments on two types of data sets. Details of these two types of data sets are presented in Appendix A.7.

- **Artificial data set:** Tables 1.4, 1.5, 1.6, 1.7, 1.8, 1.9 present results for artificial/synthetic datasets.

- **Real data set:** Tables 1.10, 1.11, 1.12 show results for real data sets.

1.4.5 Description of the rows/columns in the tables

Note that the labels for each of the columns in Tables 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 1.10, 1.11, 1.12 are as follows:

- **Case:** The first part is a name. '**Case 1**' or '**Case 2**' denotes the instance number. The second part is the format $(\text{size}, \text{cardinality})$ which denotes the number of columns/rows of the A matrix and the right-hand-side of the ℓ_0 constraint of the original SPCA problem.

- **LB:** denotes the lower bound on the SPCA problem obtained from the (heuristic) Algorithm 2 in Section 1.4.2.

- **#-λ:** denotes the size of set $\{i \mid \lambda_i > \text{LB}\}$ where λ_i are the eigenvalues of the covariance matrix. One should notice that #-λ usually does not equal to I_{pos}, since I_{pos} can be pre-determined based on threshold parameter λ_{TH}.

- **Convex-IP-ℓ_0, Pert-Convex-IP$_0$:** denote the (Convex-IP) and the (Pert-Convex-IP) models.

- **SDP:** denotes the semidefinite programming relaxation solved using Mosek. In Appendix A.9, we compare the dual bounds by alternative methods [39] to solve the SDP-relaxation for the real instances. Our conclusion based on our implementation of other algorithms is that when Mosek solves the instance, the best dual bound is obtained from Mosek. For some slightly larger instances, other algorithms might produce dual bounds. Usually, these dual bounds are extremely poor in quality. Moreover, these other methods do not scale up to instances with $d \geq 1000$. Therefore, we

have chosen to present results only from Mosek in Tables 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 1.10, 1.11, 1.12; and the remaining results are relegated to Appendix A.9.

- **UB:** denotes the upper bound obtained from current dual bound method (i.e., Convex-IP-ℓ_0, Pert-Convex-IP$_0$, SDP).
- **gap:** denotes the approximation ratio (duality gap) obtained by the formula gap $:= \frac{\text{UB}-\text{LB}}{\text{LB}}$.
- **Time:** denotes the total running time—we present the overall running time due to singular value decomposition, heuristic method to obtain primal solutions, and solvers (Gurobi, Mosek) used to solve integer programming (set to terminate within 7200 seconds).

The three rows corresponding to Pert-Convex-IP, corresponds to experiments with three settings: $(I_{\text{pos}}, N, \#\text{ iter}) = \{(5, 3, 10),\ (10, 3, 3),\ (15, 3, 2)\}$.

1.4.6 Conclusions and summary of numerical experiments

Based on numerical results reported in Tables 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 1.10, 1.11, 1.12 we draw some preliminary observations:

1. **Size of instances solved:**
   - **SDP:** Because of limitation of hardware and software, the SDP relaxation method does not solve instances with input matrix of size greater than or equal to 300×300.
   - **Convex-IP:** The convex IP shows better scalability than the SDP relaxation and produces dual bounds for instances with input matrix of size up to 500×500.
   - **Pert-Convex-IP:** The perturbed convex IP scales significantly better that the other methods. While we experimented with instances up to size 2000×2000,

we believe this method will easily scale to larger instances, when $k = 10, 20$ with (I_{pos}, N) being chosen appropriately.

2. **Quality of dual bound:**

   - **SDP vs Best of {Convex-IP, Pert-Convex-IP}:** While on some instances SDP obtained better dual bounds, this was not the case for all instances. For example, on the 'controlling sparsity' random instances and both the real data sets Eisen-1 and Eisen-2, SDP bounds are weaker.
   - **Convex-IP vs Pert-Convex-IP:** If the convex IP solved within the time limit, then usually the bound is better than that obtained for Pert-Convex-IP. In other cases, Pert-Convex-IP performs better as it is easy to solve and usually solves within 1 hour.
   - **Overall gaps for Best of {Convex-IP, Pert-Convex-IP}:** Except for the random instances of type 'controlling sparsity' of size 1000×1000, and Lymphoma data set, in all other instances at least one method had a gap less that 10%.
   - **Cardinality 10 vs Cardinality 20:** When the cardinality budget is allowed to increase, based on our numerical results, we can see that the running time of our (Convex-IP) and (Pert-Convex-IP) methods do not change a lot, since the parameter of cardinality k of (Convex-IP) and (Pert-Convex-IP) method only influences the linear constraint $\|\boldsymbol{v}\|_1 \leq \sqrt{k}$, which is more robust to changes in the value of the cardinality k than typical cardinality constraint in integer programming.

3. **Gap results under different splitting points (parameter N):** We compare the performances of the Pert-Convex-IP$_0$ method under distinct parameters of initialization splitting points with $(I_{\text{pos}}, N_{\text{ini}}, \#\,\text{iter}) = (5, 1, 1), (5, 3, 1), (5, 5, 1)$, see Table 1.1. We present results with just one round of iterations to clearly understand the effect of number of splitting points. We observe that the gap decreases when the number

of splitting points increases. On the other hand, the running time increases with the number of splitting points incereasing. However increasing splitting points from 3 to 5 does not significantly improve the bounds.

Table 1.1: Gap results under different splitting points

Name$(d,k) \setminus (I_{\text{pos}}, N_{\text{ini}}, \#\ \textbf{iter})$	**LB**	$(5,1,1)$		$(5,3,1)$		$(5,5,1)$	
		gap %	Time	gap %	Time	gap %	Time
Eisen-1 (79, 10)	17.335	2.619	2.7	0.588	3.0	**0.329**	3.1
Eisen-2 (118, 10)	11.718	13.245	5.7	4.736	7.2	**4.207**	7.8
Colon (500, 10)	2641.229	30.652	72	27.755	73	**27.673**	76
Lymphoma (500, 10)	6008.741	52.412	95	43.956	83	**43.587**	86
Reddit (2000, 10)	1052.934	8.548	1628	4.136	1450	**3.999**	1488

4. **Comparison between ℓ_1-relaxation and original sparsity constraint:** To further illustrate why we prescribe the use of ℓ_1 relaxation to obtain dual bounds of SPCA, we compare the following two models: (1) The (Pert-Convex-IP) model used in the paper; (2) The same "perturbed convex IP" where the ℓ_1 constraint is replaced by a cardinality constraint (with the introduction of binary variables), denoted as (Model-with-ℓ_0).

$$
\begin{aligned}
\max \quad & \lambda_{\text{TH}} + \textstyle\sum_{i\in I^+}(\lambda_i - \lambda_{\text{TH}})\xi_i - s =: \text{opt}_{\text{pert-convex-IP}} \\
\text{s.t.} \quad & \|\boldsymbol{v}\|_2 \le 1 \\
& \textstyle\sum_{i=1}^d \boldsymbol{z}_i \le k, \quad -\boldsymbol{z}_i \le \boldsymbol{v}_i \le \boldsymbol{z}_i, \ \ \boldsymbol{z}_i \in \{0,1\}, \ \ i \in [d] \\
& g_i = \boldsymbol{v}^\top \boldsymbol{w}_i, \quad g_i \in [-\theta_i, \theta_i], \quad i \in I^+ \\
& \begin{cases} g_i = \sum_{j=-N}^N \gamma_i^j \eta_i^j, \quad \xi_i = \sum_{j=-N}^N (\gamma_i^j)^2 \eta_i^j \\ (\eta_i^{-N}, \ldots, \eta_i^N) \in \text{SOS-2}, \quad i \in I^+ \end{cases} \qquad \text{(Model-with-}\ell_0\text{)} \\
& 1 - \tfrac{s}{\lambda_{\text{TH}} - \bar{\lambda}} \le \textstyle\sum_{i\in I^+} \xi_i \le 1 + \sum_{i\in I^+} \frac{\theta_i^2}{4N^2} - \frac{s}{\lambda_{\text{TH}} - \bar{\lambda}} \\
& \textstyle\sum_{i\in I^+} g_i^2 \le 1 - \frac{s}{\lambda_{\text{TH}} - \bar{\lambda}} \\
& \boldsymbol{c}^\top |\boldsymbol{v}| \le b_{(\boldsymbol{c})}
\end{aligned}
$$

We tested on the real-life data for $k = 10$ and $k = 20$ in Table 1.2, Table 1.3. All parameters are same as the paper that used in the Section 4.5 (except for number of

iterations which is 1 here).

Table 1.2: Gap Comparison for Real Instances with Cardinality $k = 10$

Name$(d, k) \setminus (I_{\textbf{pos}}, N_{\textbf{ini}}, \textbf{\# iter})$	**Model**	$(5, 3, 1)$		$(10, 3, 1)$		$(15, 5, 1)$	
		gap %	Time	gap %	Time	gap %	Time
Eisen-1 (79, 10)	(Pert-Convex-IP)	0.588	2.8	0.796	3.8	0.865	10
	(Model-with-ℓ_0)	**0.392**	8.6	0.525	99	0.588	685
Eisen-2 (118, 10)	(Pert-Convex-IP)	4.736	6.6	2.364	27	5.349	2610
	(Model-with-ℓ_0)	4.48	86	2.321	2105	**1.971**	5935
Matrix CovColon (500, 10)	(Pert-Convex-IP)	27.755	90	2.364	27	5.349	2610
	(Model-with-ℓ_0)	4.48	86	**2.321**	2105	11.51	7288
Matrix LymphomaCov (500, 10)	(Pert-Convex-IP)	43.956	87	23.662	355	**17.863**	4224
	(Model-with-ℓ_0)	47.93	7305	39.431	7289	39.526	7309
Reddit (2000, 10)	(Pert-Convex-IP)	4.136	1867	**3.446**	1831	3.523	3726
	(Model-with-ℓ_0)	5.826	8765	8.867	8638	10.356	8542

Table 1.3: Gap Comparison for Real Instances with Cardinality $k = 20$

Name$(d, k) \setminus (I_{\textbf{pos}}, N_{\textbf{ini}}, \textbf{\# iter})$	**Model**	$(5, 3, 1)$		$(10, 3, 1)$		$(15, 5, 1)$	
		gap %	Time	gap %	Time	gap %	Time
Eisen-1 (79, 20)	(Pert-Convex-IP)	**0.559**	3.2	0.813	20	0.886	1016
	(Model-with-ℓ_0)	1.298	7204	2.985	7204	5.519	7229
Eisen-2 (118, 20)	(Pert-Convex-IP)	1.837	6.5	1.18	46	**1.087**	443
	(Model-with-ℓ_0)	2.65	8062	4.223	7211	3.664	7205
Matrix CovColon (500, 20)	(Pert-Convex-IP)	17.014	75	6.528	372	**6.066**	7275
	(Model-with-ℓ_0)	18.539	7268	12.903	7271	12.737	7273
Matrix LymphomaCov (500, 20)	(Pert-Convex-IP)	24.042	91	14.498	214	**11.811**	3349
	(Model-with-ℓ_0)	26.622	7288	24.381	7302	35.286	8831
Reddit (2000, 20)	(Pert-Convex-IP)	**4.286**	4652	4.288	1677	4.776	4274
	(Model-with-ℓ_0)	7.139	8708	9.647	8546	12.157	8560

Based on the Table 1.2 1.3, following conclusions can be obtained:

(a) **For instances with relative small size (≤ 500):** the upper bounds (UB) obtained from (Model-with-ℓ_0) is a slightly better than the upper bounds (UB) from (Pert-Convex-IP), but the running time used for (Model-with-ℓ_0) is much longer than (Pert-Convex-IP).

(b) **For instances with relative large size (≥ 500):** both the upper bounds and the running time obtained from (Pert-Convex-IP) method are significantly better

than those obtained from (Model-with-ℓ_0). In another words, the (Pert-Convex-IP) is more scalable.

(c) **Effect of k:** We see that for $k = 20$ the performance of (Pert-Convex-IP) method is even more dramatically better than that of (Model-with-ℓ_0). In fact, now (Pert-Convex-IP) beats (Model-with-ℓ_0) on quality of bound and time even for small (≤ 500) instances. Indeed, this is another nice property of the ℓ_1-relaxation, namely it handles larger values of k more robustly.

1.4.7 Tables for numerical experiments

Table 1.4: Spiked Covariance Recovery - Cardinality 10

Case	LB	#-λ	Convex-IP-ℓ_0		Pert-Convex-IP$_0$		SDP	
			gap %	Time	gap %	Time	gap %	Time
Case 1 (200, 10)	511.95	1	0.005	380	0.007	76	**0.001**	1277
					0.005	230		
					0.005	1605		
Case 2 (200, 10)	592.45	1	0.003	469	0.006	615	**0.002**	1458
					0.006	236		
					0.005	325		
Case 1 (300, 10)	414.04	1	**0.027**	1692	0.03	642	-	-
					0.029	407		
					0.027	796		
Case 2 (300, 10)	568.56	1	**0.011**	1067	0.016	82	-	-
					0.014	493		
					0.012	942		
Case 1 (400, 10)	478.24	1	**0.025**	2598	0.04	793	-	-
					0.03	610		
					0.03	1495		
Case 2 (400, 10)	426.91	1	**0.037**	3374	0.06	181	-	-
					0.05	846		
					0.04	2137		
Case 1 (500, 10)	256.82	1	**0.164**	7525	0.21	1345	-	-
					0.18	1512		
					0.17	3279		
Case 2 (500, 10)	551.74	1	**0.029**	7196	0.04	152	-	-
					0.04	725		
					0.03	1694		
Case 1 (1000, 10)	315.16	1	-	-	0.57	1147	-	-
					0.52	776		
					0.53	3633		
Case 2 (1000, 10)	383.44	1	-	-	0.34	2745	-	-
					0.32	403		
					0.34	3643		

Table 1.5: Spiked Covariance Recovery - Cardinality 20

Case	LB	#-λ	Convex-IP-ℓ_0		Pert-Convex-IP$_0$		SDP	
			gap %	Time	gap %	Time	gap %	Time
Case 1 (200, 20)	516.756	1	2.05	493	**0.008**	746	-	-
					0.073	3116		
					0.573	7214		
Case 2 (200, 20)	593.651	1	0.98	1847	**0.005**	323	-	-
					0.006	5992		
					0.102	7215		
Case 1 (300, 20)	499.92	1	0.70	1848	**0.018**	745	-	-
					0.021	4799		
					0.399	7230		
Case 2 (300, 20)	600.553	1	1.13	1771	0.014	530	-	-
					0.013	2964		
					0.272	7232		
Case 1 (400, 20)	483.995	1	2.74	6398	**0.034**	1186	-	-
					0.168	7262		
					0.832	7255		
Case 2 (400, 20)	428.275	1	1.92	7426	**0.045**	576	-	-
					0.074	6965		
					0.53	7251		
Case 1 (500, 20)	294.35	1	1.19	7027	**0.162**	1341	-	-
					0.165	6087		
					1.285	7294		
Case 2 (500, 20)	571.15	1	1.96	4628	**0.039**	1862	-	-
					0.2	1935		
					1.215	3360		
Case 1 (1000, 20)	414	1	-	-	0.53	3133	-	-
					0.50	2760		
					0.50	5844		
Case 2 (1000, 20)	391.795	1	-	-	**0.311**	4756	-	-
					0.74	3596		
					2.906	7516		

Table 1.6: Synthetic Example - Cardinality 10

Case	LB	#-λ	Convex-IP-ℓ_0		Pert-Convex-IP$_0$		SDP	
			gap %	Time	gap %	Time	gap %	Time
Case 1 (200, 10)	5634.143	3	11.884	7205	0.14	38	**0.10**	1092
					0.15	16		
					0.15	186		
Case 2 (200, 10)	7321.23	3	1.703	7205	0.13	23	**0.09**	1086
					0.13	13		
					0.12	47		
Case 1 (300, 10)	4157.46	3	51.072	7210	**0.27**	83	-	-
					0.29	21		
					0.27	486		
Case 2 (300, 10)	5135.50	3	65.275	7210	0.23	62	-	-
					0.22	59		
					0.23	58		
Case 1 (400, 10)	6519.37	3	55.308	7219	**0.22**	98	-	-
					0.23	23		
					0.22	349		
Case 2 (400, 10)	5942.05	3	45.396	7218	**0.36**	56	-	-
					0.42	29		
					0.41	364		
Case 1 (500, 10)	5125.86	3	65.98	7230	0.38	149	-	-
					0.38	44		
					0.37	132		
Case 2 (500, 10)	5545.85	3	48.328	7230	0.39	50	-	-
					0.38	30		
					0.38	231		
Case 1 (1000, 10)	5116.08	3	NaN	-	0.58	257	-	-
					0.57	128		
					0.57	1373		
Case 2 (1000, 10)	6946.12	3	NaN	-	0.39	323	-	-
					0.36	129		
					0.34	1167		

Table 1.7: Synthetic Example- Cardinality 20

Case	**LB**	**#-λ**	**Convex-IP-ℓ_0**		**Pert-Convex-IP$_0$**		**SDP**	
			gap %	Time	gap%	Time	gap %	Time
Case 1 (200, 20)	11222.152	2	0.779	7205	**0.041**	2391	-	-
					0.042	2178		
					0.466	3707		
Case 2 (200, 20)	14588.507	2	0.503	7205	**0.032**	1285	-	-
					0.036	2772		
					0.479	7212		
Case 1 (300, 20)	8282.32	3	13.336	7212	**0.089**	2745	-	-
					0.159	1386		
					1.523	7227		
Case 2 (300, 20)	10233.583	3	4.182	7210	0.078	1835	-	-
					0.07	99		
					0.817	7229		
Case 1 (400, 20)	12976.349	3	55.172	7219	**0.08**	2563	-	-
					0.105	5278		
					4.288	7248		
Case 2 (400, 20)	11809.325	2	45.209	7219	0.082	4257	-	-
					0.084	6934		
					0.08	485		
Case 1 (500, 20)	10218.591	3	65.637	7231	**0.13**	3882	-	-
					0.142	6568		
					2.067	7288		
Case 2 (500, 20)	11032.377	3	48.034	7229	**0.114**	6603	-	-
					0.138	2753		
					4.88	7280		
Case 1 (1000, 20)	10193.919	3	-	-	1.38	303	-	-
					1.358	1707		
					0.24	3257		
Case 2 (1000, 20)	13867.929	3	-	-	0.691	318	-	-
					0.674	1927		
					0.18	8807		

Table 1.8: Controlling Sparsity - Cardinality 10

Case	LB	#-λ	Convex-IP-ℓ_0		Pert-Convex-IP$_0$		SDP	
			gap %	Time	gap %	Time	gap %	Time
Case 1 (200, 10)	706	1	**0.14**	925	2.9	117	0.42	1360
					2.6	340		
					2.6	3663		
Case 2 (200, 10)	680	1	**0.14**	1195	3.53	176	1.2	1148
					3.38	372		
					3.53	3672		
Case 1 (300, 10)	972	1	**1.4**	1958	3.91	135	-	-
					3.81	453		
					3.70	3635		
Case 2 (300, 10)	976	1	**1.1**	3007	3.79	278	-	-
					3.48	1558		
					3.69	3772		
Case 1 (400, 10)	1239	1	**1.3**	7207	4.21	769	-	-
					3.96	699		
					3.96	3699		
Case 2 (400, 10)	1207	1	**1.6**	7206	3.56	221	-	-
					3.48	1894		
					3.40	3697		
Case 1 (500, 10)	1498	1	**2.1**	12180	5.21	1026	-	-
					4.74	2881		
					4.81	3661		
Case 2 (500, 10)	1498	1	**2.1**	13917	4.14	251	-	-
					4.07	1039		
					4.01	3783		
Case 1 (1000, 10)	3948	1	-	-	59.7	2206	-	-
					53.3	8318		
					49.5	3600		
Case 2 (1000, 10)	4002	1	NaN	-	58.1	3270	-	-
					51.0	8356		
					47.6	3600		

Table 1.9: Controlling Sparsity - Cardinality 20

Case	LB	#-λ	Convex-IP-ℓ_0		Pert-Convex-IP$_0$		SDP	
			gap %	Time	gap %	Time	gap %	Time
Case 1 (200, 20)	1341.432	1	0.97	277	0.01	1434	-	-
					0.009	4726		
					0.735	2554		
Case 2 (200, 20)	1287.45	1	1.63	332	0.009	887	-	-
					0.008	2847		
					1.22	1971		
Case 1 (300, 20)	1839.578	1	1.25	1019	**0.551**	1932	-	-
					0.636	4854		
					7.027	7280		
Case 2 (300, 20)	1849.485	1	0.192	2217	**0.19**	897	-	-
					0.796	7229		
					4.287	7226		
Case 1 (400, 20)	2339.441	1	**1.45**	907	2.140	4343	-	-
					5.47	7265		
					9.847	7248		
Case 2 (400, 20)	2273.785	1	**2.34**	3106	3.572	3059	-	-
					5.864	5164		
					10.537	7249		
Case 1 (500, 20)	2870.013	1	**2.34**	2773	3.376	6013	-	-
					4.077	10870		
					5.572	7285		
Case 2 (500, 20)	2832.149	1	**2.37**	3015	3.539	5011	-	-
					5.087	7293		
					5.063	7283		
Case 1 (1000, 20)	7535.996	1	-	-	31.656	7851	-	-
					27.151	721		
					25.326	7518		
Case 2 (1000, 20)	7759.88	1	-	-	29.393	311	-	-
					25.230	809		
					23.433	7510		

Table 1.10: First six sparse principal components of Pitprops

Cardinality	**LB**	**Convex-IP-ℓ_0**		**Pert-Convex-IP**		**SDP**	
		gap %	Time	gap %	Time	gap %	Time
Cardinality 5	3.406	3.2	0.40	6.0	0.34	**1.5**	3.70
Cardinality 2	1.882	1.4	0.23	3.6	0.34	**0**	2.49
Cardinality 2	1.364	3.8	0.30	7.6	0.85	**1.0**	2.69
Cardinality 1	1	1.8	0.75	3.5	1.02	**0**	2.40
Cardinality 1	1	2.2	0.30	3.6	0.61	**0**	2.42
Cardinality 1	1	1.2	0.30	2.1	0.51	**0**	2.32
Sum of above	9.652	2.5	2.28	4.8	3.67	**0.7**	16.02

Table 1.11: Biological and Internet Data - Cardinality 10

Case	**LB**	**#-λ**	**Convex-IP-ℓ_0**		**Pert-Convex-IP$_0$**		**SDP**	
			gap %	Time	gap %	Time	gap %	Time
Eisen-1 (79, 10)	17.33	1	0.3	4.6	**0.12**	63	2.2	15
					0.17	113		
					0.4	412		
Eisen-2 (118, 10)	11.71	1	**1.4**	96	4.10	69	2.0	52
					2.13	139		
					1.70	385		
Colon (500, 10)	2641	1	14.7	9000	27.7	708	-	-
					9.58	1181		
					6.89	353		
Lymphoma (500, 10)	6008	3	20.7	3723	41	610	-	-
					21	1526		
					17	2808		
Reddit (2000, 10)	1052	1	NaN	-	3.59	5663	-	-
					2.142	8584		
					3.615	4318		

Table 1.12: Biological and Internet Data - Cardinality 20

Case	LB	#-λ	Convex-IP-ℓ_0		Pert-Convex-IP$_0$		SDP	
			gap %	Time	gap %	Time	gap %	Time
Eisen-1 (79, 20)	17.719	1	1.30	742	**0.062**	450	2.37	13
					0.102	7928		
					0.333	7205		
Eisen-2 (118, 20)	19.323	1	2.02	64	1.309	283	2.28	53
					0.502	904		
					1.294	7206		
Colon (500, 20)	4255.694	1	15.3	7230	16.537	4510	-	-
					5.77	2931		
					5.89	7286		
Lymphoma (500, 20)	9082.158	2	18.7	7239	22.569	1677	-	-
					12.3	1442		
					11.81	3721		
Reddit (2000, 20)	1119.046	1	-	-	**4.256**	7920	-	-
					4.288	1677		
					4.776	4274		

CHAPTER 2
SOLVING ROW-SPARSE PRINCIPAL COMPONENT ANALYSIS VIA CONVEX INTEGER PROGRAMS

This chapter is based on a joint work with Santanu S. Dey, Marco Molinaro, and Guanyi Wang, [40].

2.1 Introduction

In this chapter, we consider a non-trivial generalization of SPCA problem – the *row-sparse PCA* (rsPCA) problem (see, for example [41]) defined as follows: Given a sample covariance matrix $\boldsymbol{A} \in \mathbb{R}^{d\times d}$, a *sparsity parameter* k ($\leq d$), the task is to find out the top-r k-sparsity principal components $\boldsymbol{V} \in \mathbb{R}^{d\times r}$ ($r \leq k$),

$$\underset{\boldsymbol{V}^\top \boldsymbol{V}=\boldsymbol{I}^r,\ \|\boldsymbol{V}\|_0\leq k}{\arg\max} \operatorname{Tr}\left(\boldsymbol{V}^\top \boldsymbol{A}\boldsymbol{V}\right), \tag{rsPCA}$$

where the *row-sparsity constraint* $\|\boldsymbol{V}\|_0 \leq k$ denotes that there are at most k non-zero rows in matrix $\boldsymbol{V}$, i.e., the principal components share *global support*. Let

$$\mathcal{F} := \{\boldsymbol{V} \mid \boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{I}^r,\ \|\boldsymbol{V}\|_0 \leq k\}$$

denote the feasible region of rsPCA and let $\text{opt}^{\mathcal{F}}(\boldsymbol{A})$ denote the optimal value of rsPCA for sample covariance matrix $\boldsymbol{A}$.

2.1.1 Literature review

Existing approaches for solving the sparse PCA problem or its approximations can be broadly classified into the five categories.

In the first category, instead of dealing with the non-convex sparsity constraint directly, the papers [3, 4, 42, 43, 44, 45, 46, 47] incorporate additional regularizers to the objective function to enhance the sparsity of the solution. Similar to LASSO for sparse linear regression problem, these new formulations can be optimized via alternating-minimization type algorithms. We note here that the optimization problem presented in [3] is NP-hard to solve, and there is no convergence guarantee for the alternating-minimization method given in [4]. The papers [42], [43], [44], [45], [46], [47] propose their own formulations for sparse PCA problem, and show that the alternating-minimization algorithm converges to stationary (critical) points. However, the solutions obtained using the above methods cannot guarantee the row-sparsity constraint $\|\boldsymbol{V}\|_0 \leq k$. Moreover, none of these methods are able to provide worst-case guarantees.

The second category of methods work with the convex relaxations of sparsity constraint. A majority of this work is for solving rsPCA for the case where $r = 1$. The papers [21, 10, 23, 22, 48, 49] directly incorporate the sparsity constraint (for $r = 1$ case) and then relax the resulting optimization problem into some convex optimization problem — usually a semi-definite programming (SDP) relaxation. However, SDPs are often difficult to scale to large instances in practice. To be more scalable, [1] proposes a framework to find dual bounds of sparse PCA problem using convex quadratic integer program for the $r = 1$ case.

A third category of papers present fixed parameter tractable exact algorithms where the fixed parameter is usually the rank of the data matrix $\boldsymbol{A}$ and r. The paper [16] proposes an exact algorithm to find the global optimal solution of rsPCA with $r = 1$ with running-time of $O(d^{\text{rank}(\boldsymbol{A})+1} \log d)$. Later the paper [50] gives a combinatorial method for sparse PCA problem with *disjoint* supports. They show that their algorithm outputs a feasible solution within $(1 - \epsilon)$-multiplicative approximation ratio in time polynomial in data dimension d and reciprocal of ϵ, but exponential in the rank of sample covariance matrix $\boldsymbol{A}$ and r. Recently [51] provides a general method for solving rsPCA exactly with computational complexity polynomial in d, but exponential in r and $\text{rank}(\boldsymbol{A})$. The paper [51] states that

the results obtained are of theoretical nature for the low rank case, and these methods may not be practically implementable.

A fourth category of results is that of specialized iterative heuristic methods for finding good feasible solutions of rsPCA [52, 53, 12, 54, 55, 11, 16] for the $r = 1$ case. These methods do not come with worst-case guarantees. Moreover, to the best of our understanding, there is no natural way to generalize these methods for solving rsPCA when $r > 1$.

The final category of papers are those that present algorithms that perform well under the assumption of a statistical-model. Under the assumption of an underlying statistical-model, the paper [56] presents a family of estimators for rsPCA with so-called 'oracle property' via solving semidefinite relaxation of sparse PCA. The paper [57] analyzes a covariance thresholding algorithm (first proposed by [58]) for the $r = 1$ case. They show that this algorithm correctly recovers the support with high probability for sparse parameter k within order $\sqrt{M}$, with M being the number of samples. This sample complexity, combining with the lower bounds results in [59, 60], suggest that no polynomial time algorithm can do significantly better under their statistical assumptions. There are also a series of papers [41, 61, 62, 63, 64] that provide the minimax rate of estimation for sparse PCA. However, all these papers require underlying statistical models, thus do not have worst-case guarantees in the model-free case.

2.1.2 Our contributions

In this paper, we generalize the approach taken in the paper [1]. Note that this generalization is significantly non-trivial going from the case of $r = 1$ to greater values of r.

Convex relaxations of feasible region $\mathcal{F}$ (Section 2.2): Note that the objective function of rsPCA is that of maximizing a convex function. Therefore, there must be at least one extreme point of the feasible region $\mathcal{F}$ that is an optimal solution. Hence, it is important to approximate the convex hull of the feasible region well. We present two convex relaxations:

- The first convex relaxation, denoted as $\mathcal{CR}1$, uses the operator norm $\|\cdot\|_{2\to 1}$ as a proxy for row sparsity (see Section 2.1.3 for a definition). This relaxation is proven to be within a multiplicative ratio (blow up factor) of $O\left(\sqrt{\ln(r)}\right)$ of the convex hull of the feasible region $\mathcal{F}$, i.e., any point in this convex relaxation scaled down by a factor of $\approx \sqrt{\ln(r)}$ is guaranteed to be in conv($\mathcal{F}$). Thus, this result establishes that $\mathcal{CR}1$ is essentially a very good approximation of the convex hull of $\mathcal{F}$.

  To prove this result we use a novel matrix sparsification procedure that samples rows based on a weighting given by the *Pietsch-Grothendieck factorization theorem* [65]. The derivation of $\mathcal{CR}1$ and the analysis of its strength is presented in Section 2.2.1.

- Since the norm $\|\cdot\|_{2\to 1}$ is known to be NP-hard to compute [66], we also present and analyze a simpler convex relaxation of $\mathcal{F}$ which is second order cone representable, which we denote as $\mathcal{CR}2$. We show that $\mathcal{CR}2$ is within a multiplicative ratio of $O\left(\sqrt{r}\right)$ of the convex hull of the the feasible region $\mathcal{F}$. This result for $\mathcal{CR}2$ generalizes the main theoretical result in [1] for the case $r = 1$. The derivation of $\mathcal{CR}2$ and the analysis of its strength is presented in Section 2.2.2.

Upper bounding the objective function of rsPCA (Section 2.3): In order to handle the non-concavity of the objective function of rsPCA, we consider the natural approach to upper bound the objective function by piecewise linear functions which can be modeled using binary variables and special ordered sets (SOS-2) [67]. Together with the convex relaxations obtained in the previous section we arrive at a convex integer programming relaxation for rsPCA.

Moreover, we prove the following affine guarantee on the quality of the upper bound obtained by solving this convex integer program: letting $\text{ub}^{\mathcal{CR}i}$ be the optimal solution of this convex integer program using $\mathcal{CR}i$ as convex relaxation of $\mathcal{F}$, we have

$$\text{opt}^{\mathcal{F}}(\boldsymbol{A}) \leq \text{ub}^{\mathcal{CR}i} \leq \text{multiplicative-ratio-}i \cdot \text{opt}^{\mathcal{F}}(\boldsymbol{A}) + \text{additive-term}, \qquad \text{for } i \in \{1, 2\},$$

where multiplicative-ratio-$1 = O(\ln(r))$, multiplicative-ratio-$2 = O(r)$, and additive term depends on r and the parameters used in piecewise linear approximation of the objective function. In other words, the multiple term in the affine guarantee depends on the quality of the convex relaxation of the feasible region and the additive term in the affine guarantee depends on the quality of the approximation of the objective function.

New greedy algorithm (Section 2.4): We also present an efficient greedy heuristic for finding good solutions to our problem. The starting point is that we can view rsPCA as:

$$\max_{S\subseteq[d],|S|=k} f(S) \text{ where, } \quad f(S) := \left(\max_{\boldsymbol{V}\in\mathbb{R}^{d\times r} \mid \boldsymbol{V}^\top \boldsymbol{V}=\boldsymbol{I}^r, \text{supp}(\boldsymbol{V})=S} \text{Tr}\left(\boldsymbol{V}^\top \boldsymbol{A}\boldsymbol{V}\right)\right).$$

Clearly solving rsPCA reduces to the selection of the correct subset S. Therefore, it is natural to design an algorithm where we iteratively search for an improving choice of S in a neighborhood of a given value of S. A natural procedure is to remove and add one index to S in order to maximize the function f, namely move to the set

$$\tilde{S} = \text{argmax}_{T:|S\cap T|\geq k-1} f(T), \tag{2.1}$$

and repeat if $\tilde{S} \neq S$. A naive idea of solving (2.1) is by computing the objective values of all $k(d-k)$ neighborhoods supports, using eigenvalue decomposition. However, this approach is not practical. For example, if the size of the covariance matrix $d = 500$ and the sparsity parameter $k = 30$, then in each iteration, we have to compute 14100 eigenvalue decomposition of matrix of size 30×30.

Our main contribution here is to design a significantly faster heuristic by solving a proxy for (2.1). In our proposed algorithm, in each iteration instead of $k(d-k)$ eigenvalue decompositions, we will only compute one eigenvalue decomposition.

Numerical experiments (Section 2.5): Based on the above, we obtain the following "complete scheme":

- Use random and some other reasonable starts as choices of a starting support, and run the improving heuristic to produce good feasible solutions.
- Solve a convex integer program (in practice, we use $\mathcal{CR}2$ with some preprocessing of data to obtain both strength and speed, together with some other simple dimension reduction techniques) to obtain dual bounds.

Step (1) above produces good feasible solutions, and step (2) produce good dual bounds to verify the quality of the feasible solutions found in Step (1).

Numerical results are reported to illustrate the efficiency of our method (both in terms of finding good solutions and proving their high quality via dual bounds) and comparison to SDP relaxation and other benchmarks are presented.

We note that a preliminary version of this paper was published in [68]. The current version has many new results, in particular $\mathcal{CR}1$ and results on its strength are completely new, and the numerical experiments have also been completely revamped.

2.1.3 Notation

We use regular lower case letters, for example α, to denote scalars. For a positive integer n, let $[n] := \{1, \ldots, n\}$. For a set $S \subseteq \mathbb{R}^n$ and a $\rho > 0$ denote $\rho \cdot S := \{\rho x : \ x \in S\}$.

We use bold lower case letters, for example $\boldsymbol{a}$, to be vectors. We denote the i-th component of a vector $\boldsymbol{a}$ as $\boldsymbol{a}_i$. Given two vectors, $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$, we represent the inner product of $\boldsymbol{u}$ and $\boldsymbol{v}$ by $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$. Sometimes it will be convenient to represent the outer product of vectors using $\otimes$, i.e., given two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$, $\boldsymbol{a} \otimes \boldsymbol{b}$ is the matrix where $[\boldsymbol{a} \otimes \boldsymbol{b}]_{i,j} = \boldsymbol{a}_i \boldsymbol{b}_j$. We denote the unit vector in the direction of the jth coordinate as $\boldsymbol{e}^j$.

We use bold upper case letters, for example $\boldsymbol{A}$, to denote matrices. We denote the (i, j)-th component of a matrix $\boldsymbol{A}$ as $\boldsymbol{A}_{ij}$. We use supp$(\boldsymbol{A})$ to denote the support of non-zero

rows of matrix $\boldsymbol{A}$. We use regular upper case letters, for example I, to denote the set of indices. Given any matrix $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ and $I \subseteq [n], J \subseteq [m]$, we denote the sub-matrix of A with rows in I and columns in J as $\boldsymbol{A}_{I,J}$. For $I \in [m]$, to simplify notation we denote the submatrix of $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ corresponding to the rows with index in I as $\boldsymbol{A}_I$ (instead of $\boldsymbol{A}_{I,[n]}$). Similarly for $i \in [m]$, we denote the i^{th} row of $\boldsymbol{A}$ as $\boldsymbol{A}_{i,\star}$ (or $\boldsymbol{A}_i$ in short). For $J \in [n]$ again to simplify the notation, we denote the submatrix of $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ corresponding to the columns with index in J as $\boldsymbol{A}_{\star,J}$ (instead of $\boldsymbol{A}_{[m],J}$), and for $j \in [n]$, we denote the j^{th} column of $\boldsymbol{A}$ as $\boldsymbol{A}_{\star,j}$.

For a symmetric square matrix $\boldsymbol{A}$, we denote the largest eigen-value of $\boldsymbol{A}$ as $\lambda_{\max}(\boldsymbol{A})$. Given $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times n}$, two symmetric matrices, we say that $\boldsymbol{A} \preceq \boldsymbol{B}$ if $\boldsymbol{B} - \boldsymbol{A}$ is a positive semi-definite matrix. Given $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{m\times n}$, we let $\langle \boldsymbol{U}, \boldsymbol{V} \rangle = \sum_{i=1}^{m}\sum_{j=1}^{n} \boldsymbol{U}_{ij}\boldsymbol{V}_{ij}$ to be the inner product of matrices. We use $\mathbf{0}^{p,q}$ to denote the matrix of size $p \times q$ with all entries equal to zero. We use $\oplus$, as a sign of direct sum of matrices, i.e., given matrices $\boldsymbol{A} \in \mathbb{R}^{p\times q}, \boldsymbol{B} \in \mathbb{R}^{m\times n}$,

$$\boldsymbol{A} \oplus \boldsymbol{B} := \begin{pmatrix} \boldsymbol{A} & \mathbf{0}^{p,n} \\ \mathbf{0}^{m,q} & \boldsymbol{B} \end{pmatrix}.$$

The operator norm $\|\boldsymbol{A}\|_{p\to q}$ of a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ is defined as

$$\|\boldsymbol{A}\|_{p\to q} := \max_{\boldsymbol{x}\in\mathbb{R}^n, \|\boldsymbol{x}\|_p=1} \|\boldsymbol{A}\boldsymbol{x}\|_q.$$

We sometimes refer $\|\boldsymbol{A}\|_{2\to 2}$ as $\|\boldsymbol{A}\|_{\text{op}}$. Note that $\|\boldsymbol{A}\|_{\text{op}}$ is the largest singular value of $\boldsymbol{A}$. The Frobenius norm of a matrix $\boldsymbol{A}$ is denoted as $\|\boldsymbol{A}\|_F$.

2.2 Convex relaxations of $\mathcal{F}$

2.2.1 Convex relaxation 1 ($\mathcal{CR}1$)

In the vector case, i.e. $r = 1$ case, a natural convex relaxation for $\mathcal{F}$ is to control the sparsity via the ℓ_2 and ℓ_1 norms, namely to consider the set $\{\boldsymbol{v} \in \mathbb{R}^d \mid \|\boldsymbol{v}\|_2 \leq 1,\ \|\boldsymbol{v}\|_1 \leq \sqrt{k}\}$ (see [1]). It is easy to see that this is indeed a relaxation in the case $r = 1$: if $\boldsymbol{v} \in \mathcal{F}$, then by definition $\langle \boldsymbol{v}, \boldsymbol{v} \rangle = 1$ and so $\|\boldsymbol{v}\|_2 = 1$, and since $\boldsymbol{v}$ is a k-sparse vector we get, using the standard ℓ_1-vs-ℓ_2-norm comparison in k-dimensional space, $\|\boldsymbol{v}\|_1 \leq \sqrt{k} \cdot \|\boldsymbol{v}\|_2 = \sqrt{k}$. Here we consider the following generalization of this relaxation for any r:

$$\mathcal{CR}1 := \left\{ \boldsymbol{V} \in \mathbb{R}^{d \times r} :\ \|\boldsymbol{V}\|_{\text{op}} \leq 1,\ \|\boldsymbol{V}\|_{2\to 1} \leq \sqrt{k},\ \sum_{i=1}^{d} \|\boldsymbol{V}_{i,\star}\|_2 \leq \sqrt{rk} \right\}.$$

Thus we now use both the $\ell_{2\to 1}$ norm and the sum of the length of the rows of $\boldsymbol{V}$ to take the role of the ℓ_1-norm proxy for sparsity (by convexity of norms both constraints are convex). While is it not hard to see that this is a relaxation of $\mathcal{F}$, we further show that it has a provable approximation guarantee.

Theorem 2. *For every positive integers d, r, k such that $1 \leq r \leq k \leq d$ the convex relaxation $\mathcal{CR}1$ satisfies*

$$\mathcal{F} \ \subseteq\ \mathcal{CR}1 \ \subseteq\ \rho_{\mathcal{CR}1} \cdot \text{conv}\,(\mathcal{F})$$

for $\rho_{\mathcal{CR}1} = 2 + \max\{6\sqrt{2\pi},\ 18\sqrt{\log 50r}\}$. In particular $\rho_{\mathcal{CR}1} = O(\sqrt{\log r})$.

Remark 2.2.1. *One can replace in $\mathcal{CR}1$ the constraint $\|\boldsymbol{V}\|_{op} \leq 1$ by the constraint*

$$\begin{pmatrix} \boldsymbol{I}^r & -\boldsymbol{V} \\ -\boldsymbol{V} & \boldsymbol{I}^r \end{pmatrix} \succeq \boldsymbol{0},$$

which is the convex hull of the Stiefel manifold $\{\boldsymbol{V} :\ \boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{I}^r\}$ *[69].*

Proof of first inclusion in Theorem 2: $\mathcal{F} \subseteq \mathcal{CR}1$

Consider a matrix $\boldsymbol{V}$ in $\mathcal{F}$; we show that it satisfies the 3 constraints of $\mathcal{CR}1$. First, observe that

$$\begin{aligned}\|\boldsymbol{V}\|_{\text{op}} &= \max_{\boldsymbol{x}\in\mathbb{R}^n, \|\boldsymbol{x}\|_2=1} \|\boldsymbol{V}\boldsymbol{x}\|_2 \\ &= \max_{\boldsymbol{x}\in\mathbb{R}^n, \|\boldsymbol{x}\|_2=1} \sqrt{\langle \boldsymbol{V}\boldsymbol{x}, \boldsymbol{V}\boldsymbol{x}\rangle} \\ &= \max_{\boldsymbol{x}\in\mathbb{R}^n, \|\boldsymbol{x}\|_2=1} \sqrt{\langle \boldsymbol{x}, \boldsymbol{V}^\top\boldsymbol{V}\boldsymbol{x}\rangle} = 1.\end{aligned}$$

Therefore, we obtain that for $\boldsymbol{V} \in \mathcal{F}$, we have $\|\boldsymbol{V}\|_{\text{op}} \leq 1$.

For the second constraint, by definition of $\|\cdot\|_{2\to 1}$ it is equivalent to verify that $\|\boldsymbol{V}\boldsymbol{x}\|_1 \leq \sqrt{k}$ for all $\boldsymbol{x} \in \mathbb{R}^r$ such that $\|\boldsymbol{x}\|_2 \leq 1$. Since $\boldsymbol{V}$ is k-row-sparse, $\boldsymbol{V}\boldsymbol{x}$ is a k-sparse vector and hence by ℓ_1- vs ℓ_2-norm comparison in k-dim space we get $\|\boldsymbol{V}\boldsymbol{x}\|_1 \leq \sqrt{k} \cdot \|\boldsymbol{V}\boldsymbol{x}\|_2 \leq \sqrt{k}$, where the last inequality follows $\|\boldsymbol{V}\boldsymbol{x}\|_2 \leq \|\boldsymbol{V}\|_{\text{op}}$ for all $\boldsymbol{x}$ satisfying $\|\boldsymbol{x}\|_2 \leq 1$.

For the third constraint of $\mathcal{CR}1$, since $\|\boldsymbol{V}\|_{\text{op}} \leq 1$ each column of $\boldsymbol{V}$, i.e., $\boldsymbol{V}_{\star,j}$ has a 2-norm of at most 1, and since there are r columns we have:

$$r \geq \|\boldsymbol{V}\|_F^2 = \sum_{i=1}^{d} \|\boldsymbol{V}_{i,\star}\|_2^2.$$

Since V is k-row-sparse, at most k of the terms in the right-hand side is non-zero. Then again applying the ℓ_1- vs ℓ_2-norm comparison in k-dim space we get

$$\sum_{i=1}^{d} \|\boldsymbol{V}_{i,\star}\|_2 \;\leq\; \sqrt{k} \cdot \sqrt{\sum_{i} \|\boldsymbol{V}_{i,\star}\|_2^2}\,.$$

Combining the displayed inequalities gives $\sum_{i=1}^{d} \|\boldsymbol{V}_{i,\star}\|_2 \leq \sqrt{rk}$, and so the third constraint of $\mathcal{CR}1$ is satisfied.

Proof of second inclusion in Theorem 2: $\mathcal{CR}1 \subseteq \rho_{\mathcal{CR}1} \cdot \mathrm{conv}(\mathcal{F})$

We assume that $k \geq 40$, otherwise $r \leq k < 40$ and the result follows from Theorem 5. We prove the desired inclusion by comparing the support function of these sets (Proposition C.3.3.1 of [70]), namely we show that for every matrix $\boldsymbol{C} \in \mathbb{R}^{d\times r}$

$$\max_{\boldsymbol{V}\in\mathcal{CR}1} \langle \boldsymbol{C}, \boldsymbol{V}\rangle \;\leq\; \rho_{\mathcal{CR}1} \cdot \max_{\boldsymbol{V}\in\mathrm{conv}(\mathcal{F})} \langle \boldsymbol{C}, \boldsymbol{V}\rangle. \tag{2.2}$$

It will suffice to prove the following sparsification result for the optimum of the left-hand side.

Lemma 2.2.1. *Assume $k \geq 40$. Consider $\boldsymbol{C} \in \mathbb{R}^{d\times r}$ and let $\boldsymbol{V}^*$ be a matrix attaining the maximum on the left-hand side of* (2.2)*, namely $\boldsymbol{V}^* \in \arg\max_{\boldsymbol{V}\in\mathcal{CR}1}\langle \boldsymbol{C}, \boldsymbol{V}\rangle$. Then there is a matrix $\boldsymbol{V}$ with the following properties:*

1. *(Operator norm) $\|\boldsymbol{V}\|_{op} \leq 1 + \max\{\sqrt{18\pi},\, 6\sqrt{\log 50r}\}$*
2. *(Sparsity) $\boldsymbol{V}$ is k-row-sparse, namely $\|\boldsymbol{V}\|_0 \leq k$*
3. *(Value) $\langle \boldsymbol{C}, \boldsymbol{V}\rangle \geq \frac{1}{2}\langle \boldsymbol{C}, \boldsymbol{V}^*\rangle$.*

Indeed, if we have such a matrix $\boldsymbol{V}$ then $\frac{\boldsymbol{V}}{\|\boldsymbol{V}\|_{op}}$ belongs to the sparse set $\mathcal{F}$ and has value $\langle \boldsymbol{C}, \frac{\boldsymbol{V}}{\|\boldsymbol{V}\|_{op}}\rangle \geq \frac{1}{2\cdot(1+\max\{\sqrt{18\pi}, 6\sqrt{\log 50r}\})} \cdot \langle \boldsymbol{C}, \boldsymbol{V}^*\rangle$, showing that (2.2) holds.

For the remainder of the section we prove Lemma 2.2.1. The idea is to randomly sparsify $\boldsymbol{V}^*$ while controlling for operator norm and value. A standard procedure is to sample the rows of V^* with probability proportional to their squared length (see [71] for this and other sampling methods). However these more standard methods do not seem effectively leverage the information that $\|\boldsymbol{V}^*\|_{2\to 1} \leq \sqrt{k}$.

Instead, we use a novel sampling more adapted to the $\ell_{2\to 1}$-norm based on a weighting of the rows of $\boldsymbol{V}^*$ given by the so-called *Pietsch-Grothendieck factorization* [65]. We state it in a convenient form that follows by applying Theorem 2.2 of [72] to the transpose.

Theorem 3 (Pietsch-Grothendieck factorization)**.** *Any matrix $\boldsymbol{V} \in \mathbb{R}^{d\times r}$ can be factorized as $\boldsymbol{V} = \boldsymbol{W}\boldsymbol{T}$ of size $\boldsymbol{T} \in \mathbb{R}^{d\times r}$, $\boldsymbol{W} \in \mathbb{R}^{d\times d}$, where*

- *$\boldsymbol{W}$ is a nonnegative, diagonal matrix with $\sum_i \boldsymbol{W}_{ii}^2 = 1$*
- *$\|\boldsymbol{T}\|_{op} \le \sqrt{\pi/2} \cdot \|\boldsymbol{V}\|_{2\to 1}$.*

So first apply this theorem to obtain a decomposition $\boldsymbol{V}^* = \boldsymbol{W}\boldsymbol{T}$. Notice that this means the ith row of $\boldsymbol{V}^*$ is just the ith row of $\boldsymbol{T}$ multiplied by the weight $\boldsymbol{W}_{ii}$. Define the "probability"

$$p_i := \frac{k}{6}\left(\boldsymbol{W}_{ii}^2 + \frac{\|\boldsymbol{V}^*_{i,\star}\|_2}{\sum_{i'}\|\boldsymbol{V}^*_{i',\star}\|_2}\right),$$

and the truncation $\bar{p}_i = \min\{p_i, 1\}$ to make it a bonafide probability.[1] We then randomly sparsify $\boldsymbol{V}^*$ by keeping each row i with probability $\bar{p}_i$ and normalizing it: define the random matrix $\widetilde{\boldsymbol{V}} := \widetilde{\boldsymbol{W}}\boldsymbol{T}$, where $\widetilde{\boldsymbol{W}}$ is the random diagonal matrix with

$$\widetilde{\boldsymbol{W}}_{ii} := \varepsilon_i \frac{\boldsymbol{W}_{ii}}{\bar{p}_i},$$

and ε_i (the indicator that we keep row i) takes value 1 with probability $\bar{p}_i$ and 0 with probability $1 - \bar{p}_i$ (and the ε_i's are independent). Since $\mathbb{E}\widetilde{\boldsymbol{W}} = \boldsymbol{W}$ notice this is procedure is unbiased: $\mathbb{E}\widetilde{\boldsymbol{V}} = \boldsymbol{V}^*$.

We first show that $\widetilde{\boldsymbol{V}}$ satisfies each of the desired items from Lemma 2.2.1 with good probability, and then use a union bound to exhibit a matrix that proves the lemma.

[1] For some intuition: The first term parenthesis in p_i controls the variance of $\widetilde{\boldsymbol{W}}_{ii}$, which is $\mathrm{Var}(\widetilde{\boldsymbol{W}}_{ii}) \le \frac{\boldsymbol{W}_{ii}^2}{p_i} \le \frac{6}{k}$; the second term controls the largest size of a row of $\widetilde{\boldsymbol{V}}$, which is $\|\widetilde{\boldsymbol{V}}_{i,\star}\|_2 \le \|\frac{\boldsymbol{V}^*_{i,\star}}{p_i}\|_2 \le \frac{6}{k}\sum_{i'}\|\boldsymbol{V}^*_{i',\star}\|_2$, which is at most 6 because $\boldsymbol{V}^* \in \mathcal{CR}1$.

Sparsity. The number of rows $\|\widetilde{\boldsymbol{V}}\|_0$ of $\widetilde{\boldsymbol{V}}$ is precisely $\sum_{i=1}^{d} \varepsilon_i$, whose expectation is at most

$$\sum_{i=1}^{d} p_i \;=\; \frac{k}{6}\Big(\sum_i \boldsymbol{W}_{ii}^2 \;+\; 1\Big) \;=\; \frac{k}{3}.$$

Employing the multiplicative Chernoff bound (Lemma B.1.1) we get

$$\Pr\left(\|\widetilde{\boldsymbol{V}}\|_0 > k\right) \;\leq\; \left(\frac{2e}{6}\right)^k \;<\; \frac{1}{50}, \tag{2.3}$$

where the last inequality uses that $k \geq 40$.

Operator norm. Let I be the indices i where $p_i \leq 1$ (so $\bar{p}_i = p_i$), and $I^c = [d] \setminus I$ (so $\bar{p}_i = 1$ and hence $\widetilde{\boldsymbol{V}}_i = \boldsymbol{V}_i^*$). From triangle inequality we can see that $\|\widetilde{\boldsymbol{V}}\|_{\text{op}} \leq \|\widetilde{\boldsymbol{V}}_I\|_{\text{op}} + \|\widetilde{\boldsymbol{V}}_{I^c}\|_{\text{op}}$. Moreover,

$$\|\widetilde{\boldsymbol{V}}_{I^c}\|_{\text{op}} = \|\boldsymbol{V}_{I^c}^*\|_{\text{op}} \leq \|\boldsymbol{V}^*\|_{\text{op}} \leq 1,$$

where the first equality is because the rows of $\widetilde{\boldsymbol{V}}_{I^c}$ are exactly equal to the rows of $\boldsymbol{V}_{I^c}^*$ and the first inequality is because deleting rows cannot increase the operator norm, and the last inequality because $\boldsymbol{V}^* \in \mathcal{F}$. Combining these observations we get that $\|\widetilde{\boldsymbol{V}}\|_{op} \leq \|\widetilde{\boldsymbol{V}}_I\|_{op} + 1$, and so we focus on controlling the operator norm of $\widetilde{\boldsymbol{V}}_I$. We do that by applying a concentration inequality to the largest eivengalue of the PSD matrix $(\widetilde{\boldsymbol{V}}_I)^\top \widetilde{\boldsymbol{V}}_I$; the following is Theorem 1.1 of [73] plus a simple estimate (see for example page 65 of [74]).

Theorem 4. *Let $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n \in \mathbb{R}^{r\times r}$ be independent, random, symmetric matrices of dimension r. Assume with probability 1 each $\boldsymbol{X}_i$ is PSD and has largest eigenvalue $\lambda_{\max}(\boldsymbol{X}_i) \leq R$. Then*

$$\Pr\left(\lambda_{\max}\Big(\sum_i \boldsymbol{X}_i\Big) \geq \alpha\right) < r \cdot 2^{-\alpha/R}$$

for every $\alpha \geq 6\lambda_{\max}(\mathbb{E}\sum_i \boldsymbol{X}_i)$.

In the following part, to be concise, without specific description, given any matrix $\boldsymbol{V}$, for any index $i \in [d]$ or subset $I \subseteq [d]$, let $\boldsymbol{V}_i := \boldsymbol{V}_{i,\star}$ be i-th row of $\boldsymbol{V}$ and $\boldsymbol{V}_I := \boldsymbol{V}_{I,\star}$ be the submatrix of $\boldsymbol{V}$ as stated in notation.

First notice that indeed $(\widetilde{\boldsymbol{V}}_I)^\top \widetilde{\boldsymbol{V}}_I$ can be written as a sum of independent PSD matrices:

$$(\widetilde{\boldsymbol{V}}_I)^\top \widetilde{\boldsymbol{V}}_I = \sum_{i\in I} \widetilde{\boldsymbol{V}}_i \otimes \widetilde{\boldsymbol{V}}_i = \sum_{i\in I} \widetilde{\boldsymbol{W}}_{ii}^2 \, (\boldsymbol{T}_i \otimes \boldsymbol{T}_i) = \sum_{i\in I} \varepsilon_i \, \frac{\boldsymbol{W}_{ii}^2}{p_i^2} \, (\boldsymbol{T}_i \otimes \boldsymbol{T}_i). \tag{2.4}$$

To estimate the max eigenvalue of the expected matrix, $\lambda_{\max}(\mathbb{E}\,(\widetilde{\boldsymbol{V}}_I)^\top \widetilde{\boldsymbol{V}}_I)$, by definition of p_i we have $\mathbb{E}\,\varepsilon_i \frac{\boldsymbol{W}_{ii}^2}{p_i^2} \leq \frac{6}{k}$ and hence

$$\mathbb{E}\,(\widetilde{\boldsymbol{V}}_I)^\top \widetilde{\boldsymbol{V}}_I \preceq \sum_{i\in I} \frac{6}{k} (\boldsymbol{T}_i \otimes \boldsymbol{T}_i) \preceq \frac{6}{k} \sum_i (\boldsymbol{T}_i \otimes \boldsymbol{T}_i) = \frac{6}{k} \boldsymbol{T}^\top \boldsymbol{T}.$$

By the guarantee of the Pietsch-Grothendieck factorization $\|\boldsymbol{T}\|_{\text{op}} \leq \sqrt{\pi/2}\,\|\boldsymbol{V}^*\|_{2\to 1}$ and since $\boldsymbol{V}^* \in \mathcal{CR}1$ we have $\|\boldsymbol{V}^*\|_{2\to 1} \leq \sqrt{k}$, so applying these bounds to the previous displayed inequality gives

$$\lambda_{\max}\Big(\mathbb{E}\,(\widetilde{\boldsymbol{V}}_I)^\top \widetilde{\boldsymbol{V}}_I\Big) \leq \frac{6}{k} \lambda_{\max}(\boldsymbol{T}^\top \boldsymbol{T}) = \frac{6}{k} \|\boldsymbol{T}\|_{\text{op}}^2 \leq 3\pi.$$

To control the R term in Theorem 4 we look at the first equation in (2.4) and notice that for

$i \in I$

$$\begin{aligned}\lambda_{\max}\Big(\widetilde{\boldsymbol{V}}_i \otimes \widetilde{\boldsymbol{V}}_i\Big) &= \lambda_{\max}\bigg(\Big(\frac{\varepsilon_i}{p_i}\boldsymbol{V}_i^*\Big) \otimes \Big(\frac{\varepsilon_i}{p_i}\boldsymbol{V}_i^*\Big)\bigg) \\ &\le \lambda_{\max}\bigg(\frac{1}{p_i^2}(\boldsymbol{V}_i^* \otimes \boldsymbol{V}_i^*)\bigg) \\ &= \frac{\|\boldsymbol{V}_i^*\|_2^2}{p_i^2} \\ &\le \frac{36(\sum_{i'}\|\boldsymbol{V}_{i'}^*\|_2)^2}{k^2} \\ &\le 36,\end{aligned}$$

where the last inequality uses the fact $\boldsymbol{V}^* \in \mathcal{CR}1$ and hence $\sum_{i'}\|\boldsymbol{V}_{i'}^*\|_2 \le \sqrt{rk} \le k$.

Then applying Theorem 4 with $\boldsymbol{X}_i = \widetilde{\boldsymbol{V}}_i \otimes \widetilde{\boldsymbol{V}}_i$, $R = 16$ and $\alpha = \max\{6\cdot 3\pi,\, 36\log 50r\}$ we get

$$\Pr\Big(\|\widetilde{\boldsymbol{V}}_I\|_{op} \ge \sqrt{\alpha}\Big) \;=\; \Pr\Big(\lambda_{\max}((\widetilde{\boldsymbol{V}}_I)^\top \widetilde{\boldsymbol{V}}_I) \ge \alpha\Big) \;<\; \frac{1}{50}.$$

Recalling we have $\|\widetilde{\boldsymbol{V}}\|_{\text{op}} \le 1 + \|\widetilde{\boldsymbol{V}}_I\|_{\text{op}}$, this gives that

$$\|\widetilde{\boldsymbol{V}}\|_{\text{op}} > 1 + \max\{\sqrt{18\pi},\, 6\sqrt{\log 50r}\} \quad \text{happens with probability at most } \frac{1}{50}. \tag{2.5}$$

Value. We want to show that with good probability $\langle C, \widetilde{\boldsymbol{V}}\rangle \ge \frac{1}{2}\langle C, \boldsymbol{V}^*\rangle$. We use throughout the following observation: for each row i we have $\langle \boldsymbol{C}_i, \boldsymbol{V}_i^*\rangle \ge 0$, since the set $\mathcal{CR}1$ is symmetric with respect to flipping the sign of a row and V^* maximizes $\langle \boldsymbol{C}, \boldsymbol{V}^*\rangle = \sum_i \langle \boldsymbol{C}_i, \boldsymbol{V}_i^*\rangle$.

Since $\mathbb{E}\widetilde{\boldsymbol{V}} = \boldsymbol{V}^*$, we have $\mathbb{E}\langle \boldsymbol{C}_I, \widetilde{\boldsymbol{V}}_I\rangle = \langle \boldsymbol{C}_I, \boldsymbol{V}_I^*\rangle$ and

$$\begin{aligned}\text{Var}(\langle \boldsymbol{C}_I, \widetilde{\boldsymbol{V}}_I\rangle) \;&=\; \sum_{i\in I}\text{Var}(\langle \boldsymbol{C}_i, \widetilde{\boldsymbol{V}}_i\rangle) = \sum_{i\in I}\text{Var}\Big(\tfrac{\varepsilon_i}{p_i}\langle \boldsymbol{C}_i, \boldsymbol{V}_i^*\rangle\Big) \le \sum_{i\in I}\frac{\langle \boldsymbol{C}_i, \boldsymbol{V}_i^*\rangle^2}{p_i} \\ &\le\; \frac{6\sum_{i'}\|\boldsymbol{V}_{i'}^*\|_2}{k}\cdot\sum_{i\in I}\frac{\langle \boldsymbol{C}_i, \boldsymbol{V}_i^*\rangle^2}{\|\boldsymbol{V}_i^*\|_2} \;\le\; 6\cdot\Big(\max_{i\in I}\Big\langle \boldsymbol{C}_i, \tfrac{\boldsymbol{V}_i^*}{\|\boldsymbol{V}_i^*\|_2}\Big\rangle\Big)\cdot\langle \boldsymbol{C}_I, \boldsymbol{V}_I^*\rangle,\end{aligned}$$

where the second inequality uses the definition of p_i and the last inequality uses that $\sum_{i'} \|\boldsymbol{V}^*_{i'}\|_2 \leq \sqrt{rk} \leq k$ (since $\boldsymbol{V}^* \in \mathcal{CR}1$). Moreover, since $\frac{\boldsymbol{V}^*_i}{\|\boldsymbol{V}^*_i\|_2}$ also belongs to $\mathcal{CR}1$, the optimality of $\boldsymbol{V}^*$ guarantees that $\langle \boldsymbol{C}_i, \frac{\boldsymbol{V}^*_i}{\|\boldsymbol{V}^*_i\|_2} \rangle \leq \langle \boldsymbol{C}, \boldsymbol{V}^* \rangle$, and so we have the variance upper bound

$$\text{Var}(\langle \boldsymbol{C}_I, \widetilde{\boldsymbol{V}}_I \rangle) \leq 6 \cdot \langle \boldsymbol{C}, \boldsymbol{V}^* \rangle^2.$$

Using the fact that $\langle \boldsymbol{C}_{I^c}, \widetilde{\boldsymbol{V}}_{I^c} \rangle = \langle \boldsymbol{C}_{I^c}, \boldsymbol{V}^*_{I^c} \rangle$ and the one-sided Chebychev inequality (Lemma B.1.2) we get

$$\Pr\left(\langle \boldsymbol{C}, \widetilde{\boldsymbol{V}} \rangle \leq \frac{1}{2} \langle \boldsymbol{C}, \boldsymbol{V}^* \rangle \right) = \Pr\left(\langle \boldsymbol{C}_I, \widetilde{\boldsymbol{V}}_I \rangle \leq \langle \boldsymbol{C}_I, V^*_I \rangle - \frac{1}{2} \langle \boldsymbol{C}, \boldsymbol{V}^* \rangle \right) \leq \frac{6}{6 + \frac{1}{4}} = 1 - \frac{1}{25}. \tag{2.6}$$

Concluding the proof of Lemma 2.2.1. Taking a union bound over inequalities (2.3), (2.5), and (2.6), we see that with positive probability $\widetilde{V}$ satisfies all items from Lemma 2.2.1. This shows the existence of the desired matrix V and concludes the proof.

2.2.2 Convex relaxation 2 ($\mathcal{CR}2$)

Since an optimization problem involving the semi-definite constraint $\boldsymbol{V}^\top \boldsymbol{V} \preceq \boldsymbol{I}^r$ (equivalent to $\|\boldsymbol{V}\|_{op} \leq 1$) and the $\ell_{2\to 1}$-norm constraint $\|\boldsymbol{V}\|_{2\to 1} \leq \sqrt{k}$ may be challenging to solve in practice we consider the following further relaxation involving second-order cone constraints:

$$\mathcal{CR}2 := \left\{ \boldsymbol{V} \in \mathbb{R}^{d \times r} : \begin{array}{ll} \|\boldsymbol{V}_{\star,j}\|_2^2 \leq 1 & \forall j \in [r] \\ \|\boldsymbol{V}_{\star,j_1} \pm \boldsymbol{V}_{\star,j_2}\|_2^2 \leq 2 & \forall j_1 \neq j_2 \in [r] \\ \|\boldsymbol{V}_{\star,j}\|_1 \leq \sqrt{k} & \forall j \in [r] \\ \sum_{i=1}^d \|\boldsymbol{V}_i\|_2 \leq \sqrt{rk} & \end{array} \right\}.$$

This set is a relaxation of $\mathcal{CR}1$ obtained by considering the constraint $\max_{\boldsymbol{x}:\|\boldsymbol{x}\|_2\leq} \|\boldsymbol{V}\boldsymbol{x}\|_2 = \|\boldsymbol{V}\|_{op} \leq 1$ only for the vectors $\boldsymbol{x} = \boldsymbol{e}^j$ and $\boldsymbol{x} = \frac{1}{\sqrt{2}}(\boldsymbol{e}^{j_1} \pm \boldsymbol{e}^{j_2})$, and considering the constraint $\max_{\boldsymbol{x}\,|\,\|\boldsymbol{x}\|_2\leq} \|\boldsymbol{V}\boldsymbol{x}\|_1 = \|\boldsymbol{V}\|_{2\to 1} \leq \sqrt{k}$ only for the vectors $\boldsymbol{x} = \boldsymbol{e}^j$. In particular this shows that $\mathcal{CR}2$ is a relaxation of $\mathcal{CR}1$ and hence a relaxation of $\mathcal{F}$. Moreover, we show that it still gives a guaranteed approximation to this set.

Theorem 5. *For every d, r, k positive integers such that $1 \leq r \leq k \leq d$, we have*

$$\operatorname{conv}(\mathcal{F}) \subseteq \mathcal{CR}2 \subseteq \rho_{\mathcal{CR}2} \cdot \operatorname{conv}(\mathcal{F}),$$

where $\rho_{\mathcal{CR}2} \leq 1 + \sqrt{r}$.

Proof. Since we argued above that $\mathcal{CR}2$ is a relaxation of $\mathcal{F}$ it suffices to show the second inclusion $\mathcal{CR}2 \subseteq (1+\sqrt{r})\operatorname{conv}(\mathcal{F})$. So consider any $\boldsymbol{V} \in \mathcal{CR}2$, and we will show $\boldsymbol{V} \in (1+\sqrt{r})\operatorname{conv}(\mathcal{F})$.

Since the sets $\mathcal{F}$ and $\mathcal{CR}2$ are symmetric to row permutations, assume without loss of generality that the rows of $\boldsymbol{V}$ are sorted in non-decreasing length, namely $\|\boldsymbol{V}_1\|_2 \geq \|\boldsymbol{V}_2\|_2 \geq \dots$. Decompose $\boldsymbol{V}$ based on its top-k largest rows, second top-k largest rows, and so on, i.e., let $\boldsymbol{V} = \boldsymbol{V}^1 + \cdots + \boldsymbol{V}^{\lceil d/k\rceil}$ with $\boldsymbol{V}^p \in \mathbb{R}^{d\times r}$ and

$$\text{supp}(\boldsymbol{V}^1) = \{1,\dots,k\} =: I^1\,,\ \ \dots\ ,\ \text{supp}(\boldsymbol{V}^{\lceil d/k\rceil}) = \{d-(\lceil d/k\rceil-1)k,\dots,d\} =: I^m.$$

For each $p = 1,\dots,\lceil d/k\rceil$ we have $\|\|\boldsymbol{V}^p/\|\boldsymbol{V}^p\|_{\text{op}}\|_0 \leq k$ and $\|\|\boldsymbol{V}^p/\|\boldsymbol{V}^p\|_{\text{op}}\|_{\text{op}} = 1$, thus $\boldsymbol{V}^p/\|\boldsymbol{V}^p\|_{\text{op}} \in \mathcal{F}$. Observe that:

$$\boldsymbol{V} = \boldsymbol{V}^1 + \cdots \boldsymbol{V}^{\lceil d/k\rceil} = \|\boldsymbol{V}^1\|_{\text{op}}\frac{\boldsymbol{V}^1}{\|\boldsymbol{V}^1\|_{\text{op}}} + \cdots + \|\boldsymbol{V}^m\|_{\text{op}}\frac{\boldsymbol{V}^{\lceil d/k\rceil}}{\|\boldsymbol{V}^{\lceil d/k\rceil}\|_{\text{op}}} \tag{2.7}$$

$$\Rightarrow\ \frac{\boldsymbol{V}}{\sum_{p=1}^{\lceil d/k\rceil}\|\boldsymbol{V}^p\|_{\text{op}}} = \underbrace{\left(\frac{\|\boldsymbol{V}^1\|_{\text{op}}}{\sum_{p=1}^{\lceil d/k\rceil}\|\boldsymbol{V}^p\|_{\text{op}}}\right)\frac{\boldsymbol{V}^1}{\|\boldsymbol{V}^1\|_{\text{op}}} + \cdots + \left(\frac{\|\boldsymbol{V}^{\lceil d/k\rceil}\|_{\text{op}}}{\sum_{p=1}^{\lceil d/k\rceil}\|\boldsymbol{V}^p\|_{\text{op}}}\right)\frac{\boldsymbol{V}^{\lceil d/k\rceil}}{\|\boldsymbol{V}^{\lceil d/k\rceil}\|_{\text{op}}}}_{\in\operatorname{conv}(\mathcal{F})}.$$

Notice that $\|\boldsymbol{V}^1\|_{\text{op}} \leq 1$, since $\|\boldsymbol{V}\|_{\text{op}} \leq 1$ and zeroing out rows cannot increase the operator norm, and also by standard relationship between $\|\cdot\|_2$ and $\|\cdot\|_F$ we have:

$$\|\boldsymbol{V}^p\|_{\text{op}} \leq \sqrt{\sum_{i \in I^p} \|\boldsymbol{V}_i\|_2^2}.$$

Furthermore, we can bound the norm of each of these rows of $\boldsymbol{V}^p$ by the average of the rows of $\boldsymbol{V}^{p-1}$, since the rows of $\boldsymbol{V}$ are sorted in non-decreasing length. Employing these bounds we get

$$\begin{aligned}
\sum_{p=1}^{\lceil d/k \rceil} \|\boldsymbol{V}^p\|_{\text{op}} &= \|\boldsymbol{V}^1\|_{\text{op}} + \sum_{p=2}^{\lceil d/k \rceil} \|\boldsymbol{V}^p\|_{\text{op}} \\
&\leq 1 + \sum_{p=2}^{\lceil d/k \rceil} \sqrt{\left(\frac{\sum_{i \in I^{p-1}} \|\boldsymbol{V}_i\|_2}{k}\right)^2 \cdot k} \\
&= 1 + \frac{1}{\sqrt{k}} \cdot \sum_{p=2}^{\lceil d/k \rceil} \sum_{i \in I^{p-1}} \|\boldsymbol{V}_i\|_2 \\
&\leq 1 + \frac{1}{\sqrt{k}} \sum_{i=1}^{d} \|\boldsymbol{V}_i\|_2 \leq 1 + \sqrt{r} \qquad (2.8)
\end{aligned}$$

where the final inequality holds since the constraint $\sum_{i=1}^{d} \|\boldsymbol{V}_i\|_2 \leq \sqrt{rk}$ is in the description of $\mathcal{CR}2$.

Combining inequalities (2.7) and (2.8) we have

$$\boldsymbol{V} \in \left(\sum_{p=1}^{\lceil d/k \rceil} \|\boldsymbol{V}^p\|_{\text{op}}\right) \cdot \text{conv}(\mathcal{F}) \subseteq (1 + \sqrt{r}) \cdot \text{conv}(\mathcal{F}).$$

concluding the proof of the theorem. □

2.3 Upper (dual) bounds for rsPCA

Based on results in the Section 2.2, we can set-up the following optimization problem:

$$\text{opt}^{\mathcal{CR}i} := \max_{\boldsymbol{V} \in \mathcal{CR}i} \text{Tr}\left(\boldsymbol{V}^\top \boldsymbol{A}\boldsymbol{V}\right). \qquad (\mathcal{CR}i\text{-Relax})$$

The following is a straightforward Corollary of Theorem 2 and Theorem 5is:

Corollary 2.3.1. $\text{opt}^{\mathcal{F}} \leq \text{opt}^{\mathcal{CR}i} \leq \rho^2_{\mathcal{CR}i}\text{opt}^{\mathcal{F}}$ *for* $i \in \{1, 2\}$.

The challenge of solving $\mathcal{CR}i$-Relax is that the objective function is non-convex. Indeed, for $r = 1$ case, Corollary 2.3.1 provide constant multiplicative approximation ratios to rsPCA. Thus inapproximability results on solving rsPCA with $r = 1$ from [75, 14] implies that solving $\mathcal{CR}i$-Relax to optimality is NP-hard. Therefore we construct a further relaxation of the objective function.

2.3.1 Piecewise linear upper approximation of objective function

Let $\boldsymbol{A} = \sum_{j=1}^{d} \lambda_j \boldsymbol{w}_j \boldsymbol{w}_j^\top$ be the eigenvalue decomposition of sample covariance matrix $\boldsymbol{A}$ with $\lambda_1 \geq \cdots \geq \lambda_d \geq 0$. The objective function then can be represented as a summation

$$\text{Tr}\left(\boldsymbol{V}^\top \boldsymbol{A}\boldsymbol{V}\right) = \sum_{j=1}^{d} \lambda_j \sum_{i=1}^{r} (\boldsymbol{w}_j^\top \boldsymbol{v}_i)^2$$

where $\boldsymbol{v}_i$ denotes the ith column of $\boldsymbol{V}$ such that $\boldsymbol{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r)$. Set auxiliary variables $g_{ji} = \boldsymbol{w}_j^\top \boldsymbol{v}_i$ for $(j, i) \in [r] \times [d]$. Let $\boldsymbol{w}_j \in \mathbb{R}^d$ satisfy

$$|[\boldsymbol{w}_j]_{j_1}| \geq \ldots \geq |[\boldsymbol{w}_j]_{j_k}| \geq \ldots \geq |[\boldsymbol{w}_j]_{j_d}|,$$

and let

$$\theta_j = \sqrt{[\boldsymbol{w}_j]^2_{j_1} + \cdots + [\boldsymbol{w}_j]^2_{j_k}}$$

be the square root of sum of top-k largest absolute entries of $\boldsymbol{w}_j$. Since $\boldsymbol{v}_i$ is supposed to be k-sparse, it is easy to observe that g_{ji} is within the interval $[-\theta_j, \theta_j]$.

Piecewise linear approximation: To relax the non-convex objective, we can upper approximate each quadratic term g_{ji}^2 by a piecewise linear function based on a new auxiliary variable ξ_{ji} via *special ordered sets type 2* (SOS-II) constraints (PLA) as follows,

$$\mathrm{PLA}([d] \times [r]) := \left\{ (g, \xi, \eta) : \begin{array}{ll} g_{ji} = \boldsymbol{w}_j^\top \boldsymbol{v}_i & (j,i) \in [d] \times [r] \\ g_{ji} = \sum_{\ell=-N}^{N} \gamma_{ji}^{\ell} \eta_{ji}^{\ell} & (j,i) \in [d] \times [r] \\ \xi_{ji} = \sum_{\ell=-N}^{N} \left(\gamma_{ji}^{\ell}\right)^2 \eta_{ji}^{\ell} & (j,i) \in [d] \times [r] \\ \left(\eta_{ji}^{\ell}\right)_{\ell=-N}^{N} \in \text{SOS-II} & (j,i) \in [d] \times [r] \end{array} \right\}$$

where for each $(j,i) \in [d] \times [r]$, $\left(\eta_{ji}^{\ell}\right)_{\ell=-N}^{N}$ is the set of SOS-II variables, and $\left(\gamma_{ji}^{\ell}\right)_{\ell=-N}^{N}$ is the corresponding set of splitting points that satisfy:

$$\underbrace{\gamma_{ji}^{-N}}_{=-\theta_j} \leq \cdots \leq \underbrace{\gamma_{ji}^{0}}_{=0} \leq \cdots \leq \underbrace{\gamma_{ji}^{N}}_{=\theta_j}$$

which split the region $[-\theta_j, \theta_j]$ into $2N$ equal intervals. See Figure 2.1 for an example.

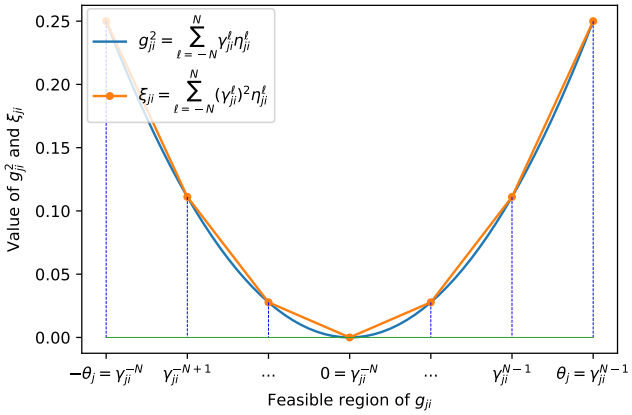


Figure 2.1: The quadratic function g_{ji}^2 is upper approximated by a piecewise linear function ξ_{ji} by SOS-II constraints for all $(j,i) \in [d] \times [r]$.

By using PLA, we arrive at the following *convex integer programming* problem,

$$\begin{aligned} \text{ub}^{\mathcal{CR}i} := \max \quad & \textstyle\sum_{j=1}^{d} \lambda_j \sum_{i=1}^{r} \xi_{ji} \\ \text{s.t.} \quad & \boldsymbol{V} \in \mathcal{CR}i \\ & (g, \xi, \eta) \in \text{PLA}([d] \times [r]) \end{aligned} \tag{CIP}$$

where $\mathcal{CR}i$ is the convex set defined in Section 2.2.1 or Section 2.2.2 for $i \in \{1, 2\}$ respectively, and PLA is the set of constraints for piecewise-linear upper approximation of objective. Note that we say this is a convex integer program since SOS-II is modelled using binary variables.

2.3.2 Guarantees on upper bounds from convex integer program

Here we present the worst-case guarantee on the upper bound from solving convex integer program in the form of an affine function of $\text{opt}^{\mathcal{F}}$.

Theorem 6. *Let $\text{opt}^{\mathcal{F}}$ be the optimal value of rsPCA. Let $\text{ub}^{\mathcal{CR}i}$ be the upper bound obtained from solving the convex integer program using $\mathcal{CR}i$ convex relaxation of $\mathcal{F}$ for $i \in \{1, 2\}$. Then:*

$$\textit{opt}^{\mathcal{F}} \leq \text{ub}^{\mathcal{CR}i} \leq \rho^2_{\mathcal{CR}i} \cdot \textit{opt}^{\mathcal{F}} + \underbrace{\sum_{j=1}^{d} \frac{r\lambda_j \theta_j^2}{4N^2}}_{\textit{additive term}}, \qquad \text{for } i \in \{1, 2\}.$$

Proof. Based on the construction for CIP, the objective function $\text{Tr}\left(\boldsymbol{V}^\top \boldsymbol{A}\boldsymbol{V}\right)$ satisfies

$$\sum_{j=1}^{d} \lambda_j \sum_{i=1}^{r} (\boldsymbol{w}_j^\top \boldsymbol{v}_i)^2 = \sum_{j=1}^{d} \lambda_j \sum_{i=1}^{r} g_{ji}^2.$$

By Corollary 2.3.1, we have

$$\max_{\boldsymbol{V} \in \mathcal{CR}i} \left(\boldsymbol{V}^\top \boldsymbol{A}\boldsymbol{V}\right) = \max_{\boldsymbol{V} \in \mathcal{CR}i} \sum_{j=1}^{d} \lambda_j \sum_{i=1}^{r} g_{ji}^2 \leq \rho^2_{\mathcal{CR}i} \cdot \text{opt}^{\mathcal{F}},$$

for $i \in \{1,2\}$. Note that $g_{ji} \in [-\theta_j, \theta_j]$ and we have split the interval $[-\theta_j, \theta_j]$ evenly via splitting points $(\gamma_{ji}^{\ell})_{\ell=-N}^{N}$ such that $\gamma_{ji}^{\ell} = \frac{\ell}{N} \cdot \theta_j$. For a given $j \in [d]$ and $i \in [r]$, by the definition of SOS-II sets, let $g_{ij} = \gamma_{ji}^{\ell^*}\eta_{j,i}^{\ell^*} + \gamma_{ji}^{\ell^*+1}\eta_{j,i}^{\ell^*+1}$, $\xi_{ji} = (\gamma_{ji}^{\ell^*})^2\eta_{j,i}^{\ell^*} + (\gamma_{ji}^{\ell^*+1})^2\eta_{j,i}^{\ell^*+1}$ and $\eta_{j,i}^{\ell^*} + \eta_{j,i}^{\ell^*+1} = 1$ for some $\ell^* \in \{-N, \ldots, N-1\}$. Thus we have:

$$\begin{aligned}
\xi_{ji} - g_{ji}^2 &= \left((\gamma_{ji}^{\ell^*})^2\eta_{j,i}^{\ell^*} + (\gamma_{ji}^{\ell^*+1})^2\eta_{j,i}^{\ell^*+1}\right) - \left(\gamma_{ji}^{\ell^*}\eta_{j,i}^{\ell^*} + \gamma_{ji}^{\ell^*+1}\eta_{j,i}^{\ell^*+1}\right)^2 \\
&= (\gamma_{ji}^{\ell^*})^2\eta_{j,i}^{\ell^*} + (\gamma_{ji}^{\ell^*+1})^2\eta_{j,i}^{\ell^*+1} - (\gamma_{ji}^{\ell^*})^2(\eta_{j,i}^{\ell^*})^2 - (\gamma_{ji}^{\ell^*+1})^2(\eta_{j,i}^{\ell^*+1})^2 - 2\gamma_{ji}^{\ell^*}\eta_{j,i}^{\ell^*}\gamma_{ji}^{\ell^*+1}\eta_{j,i}^{\ell^*+1} \\
&= \left(\gamma_{ji}^{\ell^*+1} - \gamma_{ji}^{\ell^*}\right)^2 \eta_{ji}^{\ell^*}\eta_{ji}^{\ell^*+1} = \frac{\theta_j^2}{N^2}\eta_{ji}^{\ell^*}\eta_{ji}^{\ell^*+1} \leq \frac{\theta_j^2}{4N^2}.
\end{aligned}$$

Therefore, the objective function in CIP satisfies

$$\sum_{j=1}^{d}\lambda_j\sum_{i=1}^{r}\xi_{ji} \leq \sum_{j=1}^{d}\lambda_j\sum_{i=1}^{r}g_{ji}^2 + \sum_{j=1}^{d}\frac{r\lambda_j\theta_j^2}{4N^2} \leq \rho_{\mathcal{CR}i}^2 \cdot \text{opt}^{\mathcal{F}} + \sum_{j=1}^{d}\frac{r\lambda_j\theta_j^2}{4N^2},$$

which completes the proof. □

2.4 Lower (primal) bounds for rsPCA

As mentioned in the introduction, we can view rsPCA as

$$\max_{S\subseteq[d],|S|=k} f(S) \text{ where, } \quad f(S) := \left(\max_{\boldsymbol{V}\in\mathbb{R}^{d\times r} \mid \boldsymbol{V}^\top\boldsymbol{V}=\boldsymbol{I}^r, \text{supp}(\boldsymbol{V})=S} \text{Tr}\left(\boldsymbol{V}^\top\boldsymbol{A}\boldsymbol{V}\right)\right), \tag{2.9}$$

and hence solving rsPCA reduces to selecting the correct support set S. Thus, a natural algorithm is the *1-neighborhood* local search that starts with a support set S and removes/adds one index to improve the value $f(S)$. The main issue with this strategy is that it requires an expensive eigendecomposition computation for each candidate pair i/j of indices to be removed/added in order to evaluate the function f. Here we propose a much more efficient strategy that solves a proxy version of this local search move that requires only 1 eigendecomposition per round.

For that we rewrite the problem as follows. Given a sample covariance matrix $\boldsymbol{A}$, let $\boldsymbol{A}^{1/2}$ be its positive semi-definite square root such that $\boldsymbol{A} = \boldsymbol{A}^{1/2}\boldsymbol{A}^{1/2}$. Observe that $\|\boldsymbol{A}^{\frac{1}{2}} - \boldsymbol{V}\boldsymbol{V}^\top \boldsymbol{A}^{\frac{1}{2}}\|_F^2 = \mathrm{Tr}(\boldsymbol{A}) - \mathrm{Tr}(\boldsymbol{V}^\top \boldsymbol{A}\boldsymbol{V})$, and therefore we may equivalently solve the following problem:

$$\min_{\boldsymbol{V}\in\mathbb{R}^{d\times r}} \quad \left\|\boldsymbol{A}^{1/2} - \boldsymbol{V}\boldsymbol{V}^\top \boldsymbol{A}^{1/2}\right\|_F^2 \quad \text{s.t.} \quad \boldsymbol{V}^\top\boldsymbol{V} = \boldsymbol{I}^r,\ \|\boldsymbol{V}\|_0 \le k. \qquad \text{(SPCA-alt)}$$

Therefore, SPCA-alt can be reformulated into a *two-stage (inner & outer) optimization problem*:

$$\min_{S\subseteq[d],\ |S|\le k} \quad \min_{\boldsymbol{V}_S} \quad \bar{f}(S, \boldsymbol{V}_S) \quad \text{s.t.} \quad \boldsymbol{V}_S^\top \boldsymbol{V}_S = \boldsymbol{I}^r$$

where

$$\bar{f}(S, \boldsymbol{M}) := \|(\boldsymbol{A}^{1/2})_S - \boldsymbol{M}\boldsymbol{M}^\top(\boldsymbol{A}^{1/2})_S\|_F^2 + \|(\boldsymbol{A}^{1/2})_{S^C}\|_F^2 \qquad (2.10)$$

and $S^C := [d] \setminus S$.

In order to find a solution with small $\bar{f}(S, \boldsymbol{V}_S)$ again we use a greedy swap heuristic that removes/adds one index to S. However, we avoid eigenvalue computations by keeping $\boldsymbol{M} = \boldsymbol{V}_S$ fixed and finding an improved set S' (i.e., with $\bar{f}(S', \boldsymbol{M}) \le \bar{f}(S, \boldsymbol{M})$), and only then updating the term $\boldsymbol{M}$; only the second needs 1 eigendecomposition of $\boldsymbol{A}_{S_t,S_t}$. We describe this in more detail, letting S_t and $\boldsymbol{V}_{S_t}^t$ be the iterates at round t.

Leaving Candidate: In the t-th iteration, given the iterates S_{t-1} and $\boldsymbol{V}_{S_{t-1}}^{t-1}$ from the previous iteration, for each index $j \in S_{t-1}$, let Δ_j^{out} be

$$\Delta_j^{\text{out}} := \|\boldsymbol{A}_j^{1/2}\|_2^2 - \left\|\boldsymbol{A}_{S_{t-1}}^{1/2} - \boldsymbol{V}_{S_{t-1}}\boldsymbol{V}_{S_{t-1}}^\top \boldsymbol{A}_{S_{t-1}}^{1/2}\right\|_F^2.$$

Then let $j^{\text{out}} := \arg\min_{j\in S_{t-1}} \Delta_j^{\text{out}}$ be the candidate to leave the set S_{t-1}.

Entering Candidate: Similarly, for each $j \in S_{t-1}^C$ define Δ_j^{in} as

$$\Delta_j^{\text{in}} := \|\boldsymbol{A}_j^{1/2}\|_2^2 - \left\|(\boldsymbol{A}^{1/2})_{S_{t-1}^j} - \boldsymbol{V}_{S_{t-1}}\boldsymbol{V}_{S_{t-1}}^\top(\boldsymbol{A}^{1/2})_{S_{t-1}^j}\right\|_F^2,$$

where $S_{t-1}^j := S_{t-1} - \{j^{\text{out}}\} + \{j\}$. Then let $j^{\text{in}} := \arg\max_{j \in S_{t-1}^C} \Delta_j^{\text{in}}$.

Update Rule: If $\Delta_{j^{\text{out}}}^{\text{out}} \geq \Delta_{j^{\text{in}}}^{\text{in}}$ the algorithm stops. Otherwise we perform the exchange with the candidates above, namely set $S_t = S_{t-1} - \{j^{\text{out}}\} + \{j^{\text{in}}\}$. In addition, we set $\boldsymbol{V}_{S_t}^t$ to be the minimizer of $\min\{f(S_t, \boldsymbol{M}) : \boldsymbol{M}^\top\boldsymbol{M} = \boldsymbol{I}^r\}$; for that we compute the eigendecomposition $\boldsymbol{A}_{S_t,S_t} = \boldsymbol{U}_{S_t}\boldsymbol{\Lambda}_{S_t}\boldsymbol{U}_{S_t}^\top$ of $\boldsymbol{A}_{S_t,S_t}$ and set $\boldsymbol{V}_{S_t}^t = (\boldsymbol{U}_{S_t})_{\star,[r]}$ to be the eigenvectors corresponding to top r eigenvalues. The complete pseudocode is presented in Algorithm 3.

Algorithm 3 Modified greedy neighborhood search

Input: Covariance matrix $\boldsymbol{A}$, sparsity parameter k, number of maximum iterations T **Output:** A feasible solution $\boldsymbol{V}$ for rsPCA.

Initialize with $S_0 \subseteq [d]$.
Compute eigendecomposition of A_{S_0}: $\boldsymbol{A}_{S_0,S_0} = \boldsymbol{U}_{S_0}\boldsymbol{\Lambda}_{S_0}\boldsymbol{U}_{S_t}^\top$, $\boldsymbol{V}_{S_0} = (\boldsymbol{U}_{S_0})_{\star,[r]}$
for $t = 1, \ldots, T$ **do**
    Compute the leaving candidate $j^{\text{out}} := \arg\min_{j \in S_{t-1}} \Delta_j^{\text{out}}$
    Compute the entering candidate $j^{\text{in}} := \arg\max_{j \in S_{t-1}^C} \Delta_j^{\text{in}}$
    if $\Delta_{j^{\text{in}}}^{\text{in}} > \Delta_{j^{\text{out}}}^{\text{out}}$ **then**
        Set $S_t := S_{t-1} - \{j^{\text{out}}\} + \{j^{\text{in}}\}$
        Compute the eigenvalue decomposition $(\boldsymbol{A}^{1/2})_{S_t} = \boldsymbol{U}_{S_t}\boldsymbol{\Lambda}_{S_t}\boldsymbol{U}_{S_t}^\top$
        Set $\boldsymbol{V}_{S_t}^t = (\boldsymbol{U}_{S_t})_{\star,[r]}$
    else
        Return the matrix $\boldsymbol{V}$ where in rows S_{t-1} equals $\boldsymbol{V}_{S_{t-1}}^{t-1}$ (i.e., $\boldsymbol{V}_{S_{t-1}} = \boldsymbol{V}_{S_{t-1}}^{t-1}$) and in rows S_{t-1}^C equals zero
    end if
end for

We observe that even though our procedure works only with a proxy of the original function f of the natural greedy heuristic, it still finds support sets S that monotonically decrease this objective function.

Theorem 7. *Algorithm 3 is a monotonically decreasing algorithm with respect to the objective function f, namely $f(S_t) < f(S_{t-1})$ for every iteration t.*

Proof. By optimality of $\boldsymbol{V}_{S_t}^t$ we can see that $f(S_t) = f(S_t, \boldsymbol{V}_{S_t}^t)$ for all t. Thus, letting $\boldsymbol{G}_t := \boldsymbol{I}^k - \boldsymbol{V}_{S_t}^t (\boldsymbol{V}_{S_t}^t)^\top$ to simplify the notation, we have

$$\begin{aligned} f(S_{t-1}) = f(S_{t-1}, \boldsymbol{V}_{S_{t-1}}^{t-1}) &= \left\| \boldsymbol{G}_t \, (\boldsymbol{A}^{1/2})_{S_{t-1}} \right\|_F^2 + \sum_{j \in S_{t-1}^C} \left\| (\boldsymbol{A}^{1/2})_j \right\|_2^2 \\ &= \left\| \boldsymbol{G}_t \, \boldsymbol{A}_{S_t}^{1/2} \right\|_F^2 + \sum_{j \in S_t^C} \left\| \boldsymbol{A}_j^{1/2} \right\|_2^2 + \underbrace{\Delta_{j^{\text{in}}}^{\text{in}} - \Delta_{j^{\text{out}}}^{\text{out}}}_{>0} \\ &> \left\| \boldsymbol{G}_t \boldsymbol{A}_{S_t}^{1/2} \right\|_F^2 + \sum_{j \in S_t^C} \left\| \boldsymbol{A}_j^{1/2} \right\|_2^2 \\ &= f(S_t, \boldsymbol{V}_{S_t}^t) = f(S_t). \end{aligned}$$

□

2.5 Numerical experiments

In this section we conduct computational experiments on fairly large instances to illustrate the efficiency of our proposed methods and to asses their qualities both in terms of finding good primal solutions and proving good dual bounds. We also compare our dual bound against that obtained from an SDP relaxation and from another baseline.

2.5.1 Methods for comparison

Methods for dual bounds

In order to generate dual bounds we implemented a version of our convex integer programming formulation (CIP), adding several enhancements like reduction of the number of SOS-II constraints and cutting planes in order to improve its efficiency (see [1] for related ideas for the case of $r = 1$). This implemented version is called CIP-impl, and is described in detail in Appendix B.2. For all experiments we use $N = 40$ as the level of discretization

for the objective function in CIP-impl. (For large instances we additionally use a dimension reduction technique, which we discuss later.)

We compare our proposed dual bound with the following two baselines:

- **Baseline 1:** Sum of diagonal entries of sub-matrix:

$$\text{Baseline1} := \boldsymbol{A}_{j_1,j_1} + \cdots + \boldsymbol{A}_{j_k,j_k}, \text{ where } \boldsymbol{A}_{j_1,j_1} \geq \boldsymbol{A}_{j_2,j_3} \geq \cdots \boldsymbol{A}_{j_d,j_d}.$$

  Note the sum of $\boldsymbol{A}_{j_1,j_1}, \ldots, \boldsymbol{A}_{j_k,j_k}$ is equal to sum of eigenvalues of sub-matrix indexed by $\{j_1, \ldots, j_k\}$ in $\boldsymbol{A}$, then Baseline-1 can be viewed as an upper bound for the optimal value of rsPCA. Moreover, Baseline-1 is tight when we have $r = k$.

- **Baseline 2:** The semi-definite programming relaxation:

$$\text{SDP} := \max_{\boldsymbol{P}} \ \text{Tr}\left(\boldsymbol{A}\boldsymbol{P}\right), \ \text{s.t.} \ \ \boldsymbol{I}_d \succeq \boldsymbol{P} \succeq \boldsymbol{0}, \ \text{Tr}(\boldsymbol{P}) = r, \ \boldsymbol{1}^\top |\boldsymbol{P}| \boldsymbol{1} \leq rk.$$

  Note that this is an SDP relaxation of rsPCA obtained by lifting the variables $\boldsymbol{V}$ into the product space $\boldsymbol{P} = \boldsymbol{V}\boldsymbol{V}^\top$.

Parameter for primal algorithm (lower bounds)

To obtain good feasible solutions we implemented the modified greedy neighborhood search (Algorithm 3) proposed in Section 2.4. For each instance we run this algorithm 400 times, where each time we pick the initial support set S_0 as a uniformly random subset of $[d]$ of size k. We allow a maximum of d iterations. The objective function value corresponding to the best solution from the 400 runs is declared as the lower bound.

2.5.2 Instances for numerical experiments

We conducted numerical experiments on two types of instances.

Artificial instances

These instances were generated artificially using ideas similar to that of the *spiked covariance matrix* [57] that have been used often to test algorithms in the $r = 1$ case. An instance **Artificial-**k^A is generated as follows.

We first choose a sparsity parameter $k^{\mathrm{A}} \leq \frac{d}{2}$ (which will be in the range $[30]$) and the orthonormal vectors $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ of dimension k^A given by

$$\boldsymbol{u}_1^\top = \left(\frac{1}{\sqrt{k^{\mathrm{A}}}}, \ldots, \frac{1}{\sqrt{k^{\mathrm{A}}}}\right), \qquad \boldsymbol{u}_2^\top = \left(\frac{1}{\sqrt{k^{\mathrm{A}}}}, -\frac{1}{\sqrt{k^{\mathrm{A}}}}, \ldots, \frac{1}{\sqrt{k^{\mathrm{A}}}}, -\frac{1}{\sqrt{k^{\mathrm{A}}}}, \right).$$

The *block spiked covariance matrix* $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$ is then computed as

$$\boldsymbol{\Sigma} := \boldsymbol{\Sigma}_1 \oplus \boldsymbol{\Sigma}_2 \oplus \boldsymbol{I}^{d-2k^{\mathrm{A}}},$$

where $\boldsymbol{\Sigma}_1 := 55\boldsymbol{u}_1\boldsymbol{u}_1^\top + 52\boldsymbol{u}_2\boldsymbol{u}_2^\top \in \mathbb{R}^{k^{\mathrm{A}} \times k^{\mathrm{A}}}$, $\boldsymbol{\Sigma}_2 := 50\boldsymbol{I}_{k^{\mathrm{A}}} \in \mathbb{R}^{k^{\mathrm{A}} \times k^{\mathrm{A}}}$. Finally, we sample M i.i.d. random vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M \sim N(\boldsymbol{0}_d, \boldsymbol{\Sigma})$ from the normal distribution with covariance matrix $\boldsymbol{\Sigma}$ and create the instance $\boldsymbol{A}$ as the sample covariance matrix of these vectors:

$$\boldsymbol{A} := \frac{1}{M}\left(\boldsymbol{x}_1\boldsymbol{x}_1^\top + \cdots + \boldsymbol{x}_M\boldsymbol{x}_M^\top\right).$$

In our experiments we use $d = 500$ (thus generating 500×500 matrices) and $M = 3000$ samples. Our experiments will focus on the cases $r = 2$ and $r = 3$ and we note that in these instances the optimal support set with cardinality k^{A} is different for both choices of r.

Real instances

The second type of instances are four real instances using the colon cancer dataset (CovColon) from [76], the lymphoma dataset (Lymph) from [77], and Reddit instances Reddit1500 and Reddit2000 from [1]. Table 2.1 presents the size of each instance.

Table 2.1: Real instances

name	CovColon	Lymph	Reddit1500	Reddit2000
size	500×500	500×500	1500×1500	2000×2000

2.5.3 Software & hardware

Software & Hardware: All numerical experiments are implemented on MacBookPro13 with 2GHz Intel Core i5 CPU and 8GB 1867MHz LPDDR3 Memory. The (CIP-impl) model was solved using Gurobi 7.0.2. The Baseline-2 model was solved using Mosek.

2.5.4 Performance measure

We measure the performances of CIP-impl and the baselines based on the primal-dual gap, defined as

$$\text{gap} := \frac{\text{ub} - \text{lb}}{\text{lb}}.$$

Here $\text{ub} \in \{\text{ub}^{\text{impl}}\ (\text{ub}^{\text{sub-mat}} \text{ in Section 2.5.6}), \text{Baseline-1}, \text{Baseline-2}\}$ denotes the dual bound obtained from CIP-impl or baselines. The term lb denotes the primal bound from the primal heuristic.

2.5.5 Numerical results for smaller instances

First we perform experiments on smaller instances of size 100×100. These instances were constructed by picking the submatrix corresponding to the top 100 largest diagonal entries from each instance listed in Section 2.5.2. We append a "prime" in the name of the instances to denote these smaller instances, e.g., Artificial-k^A' and CovColon'.

Time limits. We set the time limit for CIP-impl to 60 seconds and imposed no time limit on SDP. (We note that on these smaller instances SDP terminated within 600 seconds.) We

also did not impose a time limit on the primal heuristic, and just note that it took less than 120 seconds on all smaller instances.

The gaps obtained by the dual bounds using CIP-impl, Baseline1, and SDP on these instances are presented in Tables 2.2 and 2.3.

Table 2.2: Gap values for smaller artificial instances with size 100×100

name \ param: (r, k)		$(2, 10)$	$(2, 20)$	$(2, 30)$	$(3, 10)$	$(3, 20)$	$(3, 30)$
Artificial-10'	CIP-impl	**0.031**	**0.0004**	**0.0003**	**0.04**	**0.0005**	0.0004
100×100	Baseline1	3.523	4.309	4.403	2.108	2.625	2.689
	SDP	0.032	**0.0004**	**0.0003**	0.043	**0.0005**	**0.0003**
Artificial-20'	CIP-impl	0.027	**0.011**	**0.007**	**0.026**	**0.011**	**0.006**
100×100	Baseline1	3.58	7.838	8.251	2.094	4.942	5.216
	SDP	**0.02**	0.014	0.008	0.027	0.014	**0.006**
Artificial-30'	CIP-impl	0.071	0.022	**0.015**	0.074	**0.023**	**0.012**
100×100	Baseline1	3.503	7.614	11.68	2.066	4.814	7.508
	SDP	**0.03**	**0.021**	0.02	**0.051**	0.026	0.014

Table 2.3: Gap values for smaller real instances with size 100×100

name \ param: (r, k)		$(2, 10)$	$(2, 20)$	$(2, 30)$	$(3, 10)$	$(3, 20)$	$(3, 30)$
CovColon'	CIP-impl	0.12	0.119	**0.094**	0.127	0.124	0.104
100×100	Baseline1	**0.063**	**0.117**	0.132	**0.052**	**0.086**	**0.098**
	SDP	0.674	0.688	0.663	1.244	1.186	1.052
Lymp'	CIP-impl	0.329	**0.272**	**0.269**	0.225	0.296	0.32
100×100	Baseline1	**0.095**	0.277	0.392	**0.049**	**0.178**	**0.297**
	SDP	0.529	0.449	0.362	0.943	0.695	0.567
Reddit1500'	CIP-impl	**0.155**	**0.139**	**0.126**	**0.129**	**0.109**	**0.025**
100×100	Baseline1	0.695	0.396	0.99	1.197	0.811	1.294
	SDP	0.265	0.294	0.242	0.175	0.146	0.033
Reddit2000'	CIP-impl	**0.029**	**0.014**	**0.011**	**0.092**	**0.054**	**0.011**
100×100	Baseline1	0.876	1.426	1.794	0.638	1.075	1.333
	SDP	0.106	0.062	0.036	0.160	0.084	0.034

Observations:

- In Table 2.2 we see that for the relatively easy artificial instances both CIP-impl and SDP find quite tight upper bounds.

- In Table 2.3 we see that for real instances SDP is substantially dominated by both CIP-impl and Baseline1.

Overall, on the 42 instances, the dual bounds from CIP-impl are best for 28 instances, the dual bounds from Baseline-1 are best for 9 instances, and the dual bounds from SDP are best for 9 instances. Since the computation of Baseline-1 scales trivially in comparison to solving the SDP, and since SDP seems to produce dual bounds of poorer quality for the more difficult real instances — in the next section we discarded SDP from the comparison.

2.5.6 Larger instances

Sub-matrix technique for largeer instances

In order to scale the convex integer program CIP-impl to handle the larger matrices, that are now up to 2000×2000, we employ the following "sub-matrix technique" to reduce the dimension.

Given a *sub-matrix ratio parameter* $\rho_{\text{sub}} \geq 1$ satisfying $\lceil \rho_{\text{sub}} k \rceil \leq d$, let $S := \{j_1, \ldots, j_{\lceil \rho_{\text{sub}} k \rceil}\}$, where $\boldsymbol{A}_{j_1,j_1} \geq \cdots \geq \boldsymbol{A}_{j_{\lceil \rho_{\text{sub}} k \rceil}, j_{\lceil \rho_{\text{sub}} k \rceil}}$, be the index set of the top-$\lceil \rho_{\text{sub}} k \rceil$ largest diagonal entries of $\boldsymbol{A}$. Consider the blocked representation of the sample covariance matrix $\boldsymbol{A}$:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{S,S} & \boldsymbol{A}_{S,S^C} \\ \boldsymbol{A}_{S,S^C}^\top & \boldsymbol{A}_{S^C,S^C} \end{pmatrix},$$

where $S^C := [d] \setminus S$. Then the optimal value $\text{opt}^{\mathcal{F}}$ satisfies

$$\begin{aligned} \text{opt}^{\mathcal{F}} &= \max_{\boldsymbol{V} \in \mathcal{F}} \text{Tr}(\boldsymbol{V}^\top \boldsymbol{A} \boldsymbol{V}) \\ &= \max_{\boldsymbol{V} \in \mathcal{F}} \text{Tr}\left((\boldsymbol{V}_S)^\top \boldsymbol{A}_{S,S} \boldsymbol{V}_S\right) + 2\,\text{Tr}\left((\boldsymbol{V}_S)^\top \boldsymbol{A}_{S,S^C} \boldsymbol{V}_{S^C}\right) \\ &\qquad + \text{Tr}\left((\boldsymbol{V}_{S^C})^\top \boldsymbol{A}_{S^C,S^C} \boldsymbol{V}_{S^C}\right). \end{aligned} \tag{submatrix-tech}$$

The first and third term have straight forward upper bounds. Now we need to consider the

problem of finding an upper bound on $\text{Tr}\left((\boldsymbol{V}_S)^\top \boldsymbol{A}_{S,S^C}\boldsymbol{V}_{S^C}\right)$.

Let S^* be the global optimal row-support set of rsPCA. Then

$$\begin{aligned}
&\text{Tr}\left((\boldsymbol{V}_S)^\top \boldsymbol{A}_{S,S^C}\boldsymbol{V}_{S^C}\right)\\
&= \text{Tr}\left(\begin{pmatrix}(\boldsymbol{V}_{S\cap S^*})^\top & (\boldsymbol{V}_{S\setminus S^*})^\top\end{pmatrix}\begin{pmatrix}\boldsymbol{A}_{S\cap S^*,S^C\cap S^*} & \boldsymbol{A}_{S\cap S^*,S^C\setminus S^*}\\ \boldsymbol{A}_{S\setminus S^*,S^C\cap S^*} & \boldsymbol{A}_{S\setminus S^*,S^C\setminus S^*}\end{pmatrix}\begin{pmatrix}\boldsymbol{V}_{S^C\cap S^*}\\ \boldsymbol{V}_{S^C\setminus S^*}\end{pmatrix}\right)\\
&= \text{Tr}\left((\boldsymbol{V}_{S\cap S^*})^\top \boldsymbol{A}_{S\cap S^*,S^C\cap S^*}\boldsymbol{V}_{S^C\cap S^*}\right).
\end{aligned}$$

Since $\boldsymbol{V}^\top\boldsymbol{V} = \boldsymbol{I}^r$, then we have $\boldsymbol{V}_{S\cap S^*}^\top\boldsymbol{V}_{S\cap S^*} + \boldsymbol{V}_{S^C\cap S^*}^\top\boldsymbol{V}_{S^C\cap S^*} = \boldsymbol{I}^r$. Thus it is sufficient to consider the following optimization problem:

$$2\max_{\boldsymbol{V}^1,\boldsymbol{V}^2} \text{Tr}\left((\boldsymbol{V}^1)^\top \boldsymbol{A}_{S\cap S^*,S^C\cap S^*}\boldsymbol{V}^2\right) \text{ s.t. } (\boldsymbol{V}^1)^\top\boldsymbol{V}^1 + (\boldsymbol{V}^2)^\top\boldsymbol{V}^2 = \boldsymbol{I}^r,$$

We show in Proposition B.2.2, proved in the appendix, that the above term is upper bounded by $\sqrt{r}\cdot\|\boldsymbol{A}_{(S\cap S^*),(S^C\cap S^*)}\|_F$.

Therefore, letting $\tilde{k} := |S\cap S^*|$ be the cardinality of the intersection, we can upper bound the right-hand side of (submatrix-tech) as

$$\text{opt}^{\mathcal{F}} \le \text{ub}^{\text{CIP}}(\boldsymbol{A}_{S,S};\tilde{k}) + \sqrt{r}\cdot\|\boldsymbol{A}_{S\cap S^*,S^C\cap S^*}\|_F + \text{Baseline-1}(\boldsymbol{A}_{S^C,S^C};k-\tilde{k}),$$

where the first term $\text{ub}^{\text{CIP}}(\boldsymbol{A}_{S,S};\tilde{k})$ is the optimal value obtained from CIP-impl with covariance matrix $\boldsymbol{A}_{S,S}$ and sparsity parameter $\tilde{k}$ (if $\tilde{k} < r$, then reset $\tilde{k} = r$), and the the third term is the value of Baseline-1 obtained from $\boldsymbol{A}_{S^C,S^C}$ with sparsity parameter $k-\tilde{k}$.

Since S^* is unknown, then the second term can be further upper bounded by

$$\|\boldsymbol{A}_{S\cap S^*,S^*\setminus S}\|_F \le \sqrt{\left\|\boldsymbol{A}_{\{j_1\},S^C}^{k-\tilde{k}}\right\|_2^2 + \cdots + \left\|\boldsymbol{A}_{\{j_{\tilde{k}}\},S^C}^{k-\tilde{k}}\right\|_2^2} =: \text{ub}(S;\tilde{k};S^C;k-\tilde{k}),$$

where

$$\|\boldsymbol{A}^l_{\{j\},S^C}\|_2^2 := \boldsymbol{A}^2_{j,i_1} + \cdots + \boldsymbol{A}^2_{j,i_l} \text{ with } |\boldsymbol{A}_{j,i_1}| \geq \cdots \geq |\boldsymbol{A}_{j,i_l}| \geq \ldots \text{ for all } i \in S^C,$$

and $j_1, \ldots, j_{\tilde{k}}$ are indices satisfying: $\left\|\boldsymbol{A}^{k-\tilde{k}}_{j_1,S^C}\right\|_2^2 \geq \cdots \geq \left\|\boldsymbol{A}^{k-\tilde{k}}_{j_{\tilde{k}},S^C}\right\|_2^2 \geq \cdots$.

Since $\tilde{k}$ is also not known, we arrive at our final upper bound $\text{ub}^{\text{sub-mat}}$ by considering all of its possibilities:

$$\text{opt}^{\mathcal{F}} \leq \underbrace{\max_{\tilde{k}=0}^{k} \left\{ \text{ub}^{\text{CIP}}(\boldsymbol{A}_{S,S}; \tilde{k}) + \sqrt{r} \cdot \text{ub}(S; \tilde{k}; S^C; k - \tilde{k}) + \text{Baseline-1}(\boldsymbol{A}_{S^C,S^C}; k - \tilde{k}) \right\}}_{=:\text{ub}^{\text{sub-mat}}}.$$

Times for larger instances

We set a more stringent time limit of 20 seconds for each CIP-impl used within the sub-matrix technique, since a number of these computations are required to compute $\text{ub}^{\text{sub-mat}}$. Again we did not set a time limit for the primal heuristic, an just note its running times as a function of the matrix size on Table 2.4.

size	500×500	1500×1500	2000×2000
running time	≤ 20 min	≤ 100 min	≤ 120 min

Table 2.4: Running time for primal heuristic

Results on larger instances

We compare the gap obtained by the upper bound $\text{ub}^{\text{sub-mat}}$ (CIP-impl plus sub-matrix technique) and compare it against that obtained by Baseline1 on the artificial and real instances with original sizes. These are reported on Tables 2.5 and 2.6.

On the spiked covariance matrix artificial instances we see that our dual bound $\text{ub}^{\text{sub-mat}}$ is typically orders of magnitude better than Baseline1, and is at most 0.35 for all instances. These results also illustrate that the sub-matrix ratio parameter can have a big impact on

the bound obtained by the sub-matrix technique.

On the real instances, we see from Table 2.6 that on instances CovColon and Lymph our dual bound $\text{ub}^{\text{sub-mat}}$ performs slightly better than Baseline1, and the gaps are overall less than 0.45 (except instance Lymph with parameters $(2, 50)$). However, on instances Reddit1500 and Reddit2000 our dual bound $\text{ub}^{\text{sub-mat}}$ vastly outperforms Baseline1 on all settings of parameters. We remark that these are the largest instances in the experiments, which attest the scalability of our proposed bound.

Table 2.5: Gap values for artificial instances.

name \ param: (r, k)		$(2, 10)$	$(2, 20)$	$(2, 30)$	$(3, 10)$	$(3, 20)$	$(3, 30)$
Artificial-10	$\rho_{\text{sub}} = 1.5$	0.527	0.151	0.25	0.366	0.1	0.169
500×500	$\rho_{\text{sub}} = 2$	0.079	0.15	0.249	0.064	0.1	0.169
	$\rho_{\text{sub}} = 2.5$	0.079	0.15	0.248	0.064	0.099	0.168
	$\rho_{\text{sub}} = 5$	0.071	0.145	0.241	0.056	0.099	0.293
	$\rho_{\text{sub}} = 10$	**0.026**	**0.002**	**0.002**	**0.03**	**0.003**	**0.003**
	Baseline1	3.522	4.309	4.403	2.101	2.625	2.688
Artificial-20	$\rho_{\text{sub}} = 1.5$	2.397	0.566	0.268	1.629	0.384	0.186
500×500	$\rho_{\text{sub}} = 2$	0.455	0.179	0.266	0.317	0.127	0.185
	$\rho_{\text{sub}} = 2.5$	0.606	0.178	0.265	0.463	0.126	0.184
	$\rho_{\text{sub}} = 5$	0.097	0.176	0.261	**0.078**	0.124	0.346
	$\rho_{\text{sub}} = 10$	**0.073**	**0.014**	**0.009**	0.139	**0.013**	**0.008**
	Baseline1	3.58	7.838	8.251	2.097	4.942	5.216
Artificial-30	$\rho_{\text{sub}} = 1.5$	3.515	0.595	0.65	2.071	0.406	0.425
500×500	$\rho_{\text{sub}} = 2$	3.509	0.721	0.314	2.068	0.512	0.211
	$\rho_{\text{sub}} = 2.5$	2.304	0.709	0.312	1.586	0.511	0.209
	$\rho_{\text{sub}} = 5$	0.474	0.225	0.305	0.365	0.158	0.468
	$\rho_{\text{sub}} = 10$	**0.231**	**0.026**	**0.017**	**0.349**	**0.154**	**0.014**
	Baseline1	3.519	7.626	11.68	2.074	4.82	7.508

2.6 Conclusion

In this paper, we proposed a scheme for producing good primal feasible solutions and dual bounds for rsPCA problem. The primal feasible solution is obtained from a monotonically improving heuristic for rsPCA problem. We showed that the solution produced by this al-

Table 2.6: Gap values for real instances.

name \ para: (r,k)		$(2,10)$	$(2,20)$	$(2,30)$	$(3,10)$	$(3,20)$	$(3,30)$
CovColon	$\rho_{\text{sub}}=1.5$	0.054	0.112	0.128	0.05	0.08	0.092
500×500	$\rho_{\text{sub}}=2$	0.051	0.107	0.126	0.062	**0.076**	0.09
	$\rho_{\text{sub}}=2.5$	**0.05**	**0.104**	**0.124**	0.066	0.089	**0.088**
	$\rho_{\text{sub}}=5$	0.094	0.113	0.143	0.11	0.122	2.349
	$\rho_{\text{sub}}=10$	1.787	1.709	1.645	3.321	3.124	3.015
	Baseline1	0.063	0.118	0.133	**0.049**	0.086	0.097
Lymph	$\rho_{\text{sub}}=1.5$	0.09	0.27	0.41	0.064	0.174	0.315
500×500	$\rho_{\text{sub}}=2$	**0.078**	0.267	0.406	0.103	**0.171**	0.312
	$\rho_{\text{sub}}=2.5$	0.104	**0.264**	0.403	0.155	0.194	**0.309**
	$\rho_{\text{sub}}=5$	0.236	0.268	**0.388**	0.2	0.296	2.698
	$\rho_{\text{sub}}=10$	2.105	1.738	1.548	4.489	3.894	3.447
	Baseline1	0.095	0.277	0.413	**0.049**	0.18	0.319
Reddit1500	$\rho_{\text{sub}}=1.5$	0.687	0.95	0.8	0.39	0.625	0.677
1500×1500	$\rho_{\text{sub}}=2$	0.683	0.94	0.749	0.387	0.617	0.632
	$\rho_{\text{sub}}=2.5$	0.672	0.937	**0.727**	0.377	0.614	**0.611**
	$\rho_{\text{sub}}=5$	0.426	**0.47**	1.068	0.346	**0.393**	1.307
	$\rho_{\text{sub}}=10$	**0.384**	0.927	1.075	**0.316**	1.222	1.343
	Baseline1	0.695	0.962	1.199	0.396	0.635	0.848
Reddit2000	$\rho_{\text{sub}}=1.5$	0.845	1.408	0.76	0.556	1.026	0.667
2000×2000	$\rho_{\text{sub}}=2$	0.837	1.4	0.664	0.549	1.019	0.585
	$\rho_{\text{sub}}=2.5$	0.827	1.396	**0.601**	0.541	1.016	**0.538**
	$\rho_{\text{sub}}=5$	0.456	**0.436**	1.52	0.395	**0.381**	1.311
	$\rho_{\text{sub}}=10$	**0.298**	0.866	2.234	**0.266**	1.289	1.41
	Baseline1	0.876	1.426	1.775	0.582	1.041	1.326

gorithm are of very high quality by comparing the objective value of the solutions generated to upper bounds. These upper bounds are obtained using second order cone IP relaxation designed in this paper. We also presented theoretical guarantees (affine guarantee) on the quality of the upper bounds produced by the second order cone IP. The running-time for both the primal algorithm and the dual bounding heuristic are very reasonable (less than 2 hours for the 500×500 instances and less than 3.5 hours for the 2000×2000 instance). These problems are quite challenging and on some instances, we still need more techniques to close the gap. However, to the best of our knowledge, there is no comparable theoretical or computational results for solving model-free rsPCA.

CHAPTER 3

APPROXIMATION ALGORITHMS FOR TRAINING ONE-NODE RELU NEURAL NETWORKS

This chapter is based on a joint work with Santanu S. Dey, Guanyi Wang, and Yao Xie, [78].

3.1 Introduction

Training neural networks with the ReLU activation function is a very important problem in machine learning. One common practice is to minimize the ℓ_2-norm empirical risk loss. Among all possible neural networks, the simplest version is the one-node (single neuron) neural network with the ReLU activation function. Thus, as a first step towards understanding the theoretical properties of this fundamental problem, we study the training of the basic neural network: a single node with the rectified linear unit function (ReLU) as its activation function (see Figure 3.1). Formally, we consider the following problem. Given a set of n training samples $\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n \in \mathbb{R}^d \times \mathbb{R}$, where $\boldsymbol{X}_i$ denotes the i^{th} input sample, and Y_i denotes the i^{th} output sample, the task is to minimize the empirical risk function defined as follows:

$$\min_{(\boldsymbol{\beta},\beta_0)\in\mathbb{R}^d\times\mathbb{R}} \frac{1}{n}\sum_{i=1}^{n} \left(\max\{\boldsymbol{X}_i^\top\boldsymbol{\beta}+\beta_0, 0\} - Y_i\right)^2, \qquad \text{(ReLU-regression)}$$

to be concise, let

$$\boldsymbol{X} := \begin{pmatrix} - & \boldsymbol{X}_1^\top & - \\ & \vdots & \\ - & \boldsymbol{X}_n^\top & - \end{pmatrix} \in \mathbb{R}^{n\times d}, \quad \boldsymbol{Y} := \begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix} \in \mathbb{R}^n,$$

the above optimization problem ReLU-regression can be represented as

$$\min_{(\boldsymbol{\beta},\beta_0)\in\mathbb{R}^d\times\mathbb{R}} \frac{1}{n}\|\max\{\boldsymbol{X}\boldsymbol{\beta}+\beta_0\mathbf{1},\mathbf{0}\}-\boldsymbol{Y}\|_2^2.$$

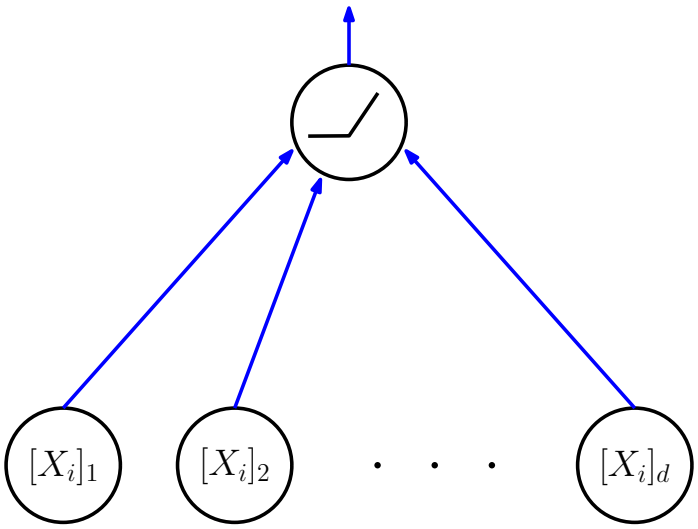


Figure 3.1: Single node neural network with ReLU activate function.

The neural network training problem, ReLU-regression, can be viewed as performing nonlinear regression, which is the perspective shared by seminal works [79, 80, 81]. A series of existing literatures [80, 82, 83, 84, 85] also study this problem from other perspectives. However, it remains an intriguing question regarding what is the *algorithmic complexity* of training neural networks from data, and if it is NP-hard (which we prove in this paper), whether one can find a good polynomial-time approximate algorithm with good approximation ratio? Whether we can improve the widely used heuristic algorithms using insights gained from developing polynomial time complexity algorithms? Partial answers to the above questions have been provided in this regard in the literature (which we will discuss and contrast as we develop our theoretical results). However, complete answers are yet to be revealed. One caveat we would like to remind readers is that the phrase *complexity of training a neural network* has been used in various contexts, for instance:

- In [86], the goal is to achieve a reliable (agnostic) learning of the ReLU neural network, i.e., finding a feasible solution that satisfies the constraints on false-positive

rates and the so-called expected loss.

- In [87], two metrics are considered: the *sampling complexity*, the number of samples needed to learn a particular class of function; and the *statistical query complexity*, the number of queries that any statistical algorithm needs to achieve an error tolerance, which is inversely proportional to the input dimensionality.

Note that all the above metrics are different from algorithmic or computational complexity.

In this paper, we aim to answer the fundamental question of *computational complexity* for training neural networks. Here, the term computational complexity means the amount of computational effort needed to solve the related optimization problem ReLU-regression, and therefore developing efficient algorithms (in the sense of computational complexity) with worse-case guarantees. We show that training a one-node ReLU network by solving ReLU-regression is NP-hard. Besides showing the NP-hardness, we also present a polynomial-time approximation algorithm with a performance guarantee to solve the problem. Key features of our results are:

1. We present a polynomial-time approximation algorithm to solve ReLU-regression via its convex approximations. This is in contrast to most existing results [88, 89, 90, 91, 92, 93, 94, 95, 85, 96, 84, 97, 98, 83, 81], which study the gradient descent (GD), stochastic gradient descent (SGD) or their variants, and show those algorithms can find locally or globally optimal solutions under some ground truth statistical model.

2. Our algorithm comes with performance guarantees for arbitrary data. Most results in the literature provide additive guarantees, in comparison to the multiplicative approximation guarantee that we can show for our algorithm.

3. Under reasonable statistical models for data, we show that the approximation ratio of our algorithm can be improved dramatically to a constant factor by removing the scaling dependence on the sample size and the dimension (i.e., independent of sample size and the dimension).

4. Our approximate algorithm does not require special initialization, (for example, the tensor initialization in [99] and the randomized initialization in [100]); we also do not need special initialization when proving the theoretical results.

5. We present extensive numerical comparisons with existing results. Our results show that our algorithm can also be utilized as a good initialization for GD/SGD based methods and achieve a significant performance gain than just using GD/SGD with random initialization.

An interesting ingredient of our proof is that we find solving ReLU-regression can be viewed as a variant of the classic *best subset selection* problem in statistics [101, 102, 103]. We refer to it as *active subset selection*, which also motivates us to develop the approximation algorithm.

The rest of the paper is organized as follows. Section 3.2 presents our theoretical results and highlights comparison with related results in the literature. Section 3.3 presents numerical results. Section 3.4 concludes the paper with discussions. Section C contains all proofs.

3.2 Theoretical Results

Training a one-node neural network as defined in ReLU-regression is a non-convex optimization problem, which is expected to be challenging to solve. However, not all non-convex problems are "difficult" (i.e., NP-hard): for example, the classical principal component analysis problem is non-convex but can be solved in polynomial-time.

Here, we analyze the optimization problem ReLU-regression in two scenarios:

1. Arbitrary data (model-free). In this case, we do not make any assumption about training sample. We would like to find optimal values of $\boldsymbol{\beta}, \beta_0$ for the minimization problem ReLU-regression. We want to answer the following questions: Is this problem NP-hard? Are there good approximation algorithms and how well they perform

in the worst scenarios?

2. Assuming an underlying statistical model for data. In this case, we assume that the training sample is of the form: (1) $\boldsymbol{X}_i$'s are *i.i.d.* sampled from a "reasonable" distribution, (2) $Y_i = \max\{0, \boldsymbol{X}_i^\top \boldsymbol{\beta}^* + \beta_0^*\} + \epsilon$, where ϵ is a Gaussian noise and $(\boldsymbol{\beta}^*, \beta_0^*)$ being the true parameters. We show that the same approximation algorithm described above for arbitrary data also works well in this case.

3.2.1 Training ReLU-regression With Arbitrary Data

Given an arbitrary fixed sample set $\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n$, we study ReLU-regression in terms of the computational complexity.

Our first result formalizes the fact that we expect solving ReLU-regression to be challenging.

Theorem 8 (NP-hardness)**.** *The ReLU-regression problem is NP-hard.*

Insight for Theorem 8. The NP-hardness result is shown by proving that the subset sum problem can be polynomially reduced to the ReLU-regression problem. The main technique is constructing two types of quadratic auxiliary function whose global minima can be used to obtain a feasible solution for any given subset sum problem or to show that a given subset sum problem is infeasible. See Appendix C.2 for a proof.

Comparison with related results in the literature. We study training ReLU neural networks from the perspective of NP-hardness when the input data are fixed and given, whereas

- [104] studies a problem of training two-layer $(d+1)$ nodes neural network with ReLU as the activation and shows that the training problem is NP-hard. Here, we show that an even more simplified structure, namely, a neural network with one node, is NP-hard.

- There are some recent works have also studied the theoretical aspects of learning ReLUs with simple structure [82, 86, 104]. Moreover, [84] independently gives another NP-hardness reductions. [1]

Based on NP-hardness result in Theorem 8, it is natural to seek an efficient approximation algorithm with multiplicative performance bound. We first introduce some basic notions that explain the design of the algorithm. Note that in ReLU-regression, we do not assume $Y_i \geq 0$ holds for every $i \in [n]$. Under this formulation, if there exists i such that $Y_i < 0$, then the optimal objective function cannot be 0. Without loss of generality, we assume that the index set with respect to all positive output samples has indices from 1 to m, i.e., $\mathcal{I}^+ := \{i \in [n] : Y_i > 0\} = \{1, \ldots, m\}$ $(= [m])$, and the index set with respect to all non-positive output samples is $\mathcal{I}^- := \{i \in [n] : Y_i \leq 0\} = \{m+1, \ldots, n\}$. We can represent the ReLU-regression problem by writing the summation as two parts:

$$\begin{aligned} &\min_{(\beta,\beta_0)\in\mathbb{R}^d\times\mathbb{R}} \| \max\{\mathbf{0}, \boldsymbol{X}\boldsymbol{\beta} + \beta_0\mathbf{1}\} - \boldsymbol{Y}\|_2^2 \\ = &\min_{(\beta,\beta_0)\in\mathbb{R}^d\times\mathbb{R}} \sum_{i\in\{1,\ldots,m\}} (\max\{0, \boldsymbol{X}_i^\top\boldsymbol{\beta} + \beta_0\} - Y_i)^2 + \phi(\boldsymbol{\beta}, \beta_0) \end{aligned} \tag{3.1}$$

where $\phi(\boldsymbol{\beta}, \beta_0) := \sum_{i\in\{m+1,\ldots,n\}}(\max\{0, \boldsymbol{X}_i^\top\boldsymbol{\beta} + \beta_0\} - Y_i)^2$ is the loss contributed from samples with non-positive responses. We can readily verify that:

Proposition 3.2.1. *The second term of (3.1), $\phi(\boldsymbol{\beta}, \beta_0)$, is convex.*

The first term of (3.1) can be further represented as a two-phase optimization problem as follows:

$$\begin{aligned} &\min_{(\beta,\beta_0)\in\mathbb{R}^p\times\mathbb{R}} \| \max\{\mathbf{0}, \boldsymbol{X}\boldsymbol{\beta} + \beta_0\mathbf{1}\} - \boldsymbol{Y}\|_2^2 \\ = &\left(\min_{I\subseteq[m]} \left(\min_{(\boldsymbol{\beta},\beta_0)\in P(I)} f_I(\boldsymbol{\beta}, \beta_0)\right)\right) + \phi(\boldsymbol{\beta}, \beta_0). \end{aligned}$$

[1] Note that [84] proposed a different approach for the hardness reduction. Moreover, the publication date of [84] on arXiv is later than the time that we submitted the arXiv version of our paper. We cite paper [84] for completeness.

Note that for any given subset $I \subseteq [m]$, i.e., $\mathcal{I}^+$, we use $P(I)$ to denote the feasible region of $\boldsymbol{\beta}$, and use $f_I(\boldsymbol{\beta}, \beta_0)$ to denote the objective function corresponding to $P(I)$ as follows

$$P(I) := \left\{ (\boldsymbol{\beta}, \beta_0) : \begin{array}{ll} \boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0 > 0, & i \in I \\ \boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0 \leq 0, & i \in [m] \backslash I \end{array} \right\},$$

$$f_I(\boldsymbol{\beta}, \beta_0) := \sum_{i \in I} (\boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0 - Y_i)^2 + \sum_{i \in [m] \backslash I} Y_i^2.$$

Henceforth, let $I \subseteq [m]$ be the *active set* where for each $i \in I$, we have the corresponding sample $(\boldsymbol{X}_i, \beta_0)$ satisfies $\boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0 > 0$, and define the index set $I^C = [m] \backslash I$ be the *inactive set*. The original ReLU-regression problem can be interpreted as a two-phase optimization problem. For any given $I \subseteq [m]$, the inner-phase optimization problem

$$z^*(I) := \min_{(\boldsymbol{\beta}, \beta_0) \in P(I)} f_I(\boldsymbol{\beta}, \beta_0) + \phi(\boldsymbol{\beta}, \beta_0),$$

is convex over $(\boldsymbol{\beta}, \beta_0)$. We will show that only a polynomial (rather than exponentially) number of distinct I's need to be examined, which is a basis of our approximation algorithm.

To obtain the approximation guarantees, we first consider an "unconstrained version" of the optimization problem corresponding to $z^*(I)$, for the ease of presentation. Define $\sigma : \mathbb{R} \times \mathbb{R}_+ \mapsto \mathbb{R}$:

$$\sigma(x, y) = \begin{cases} (x - y)^2 & \text{if } x > 2y \\ y^2 & \text{if } x \leq 2y, \end{cases}$$

where

$$\sigma(\boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0, Y_i) \begin{cases} = (\max\{0, \boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0\} - Y_i)^2 & \text{if } \boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0 \leq 0 \text{ or } \boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0 \geq 2Y_i, \\ > (\max\{0, \boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0\} - Y_i)^2 & \text{otherwise.} \end{cases}$$

σ is convex, as illustrated in Figure 3.2. Let

$$f_I^\sigma(\boldsymbol{\beta}, \beta_0) := \sum_{i \in I} (\boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0 - Y_i)^2 + \sum_{i \in [m] \backslash I} \sigma(\boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0, Y_i)$$

be a convex upper approximation of $f_I(\boldsymbol{\beta}, \beta_0)$. Thus,

$$z^\sigma(I) := \min_{(\boldsymbol{\beta}, \beta_0) \in \mathbb{R}^d \times \mathbb{R}} f_I^\sigma(\boldsymbol{\beta}, \beta_0) + \phi(\boldsymbol{\beta}, \beta_0)$$

is a convex upper approximation of $z^*(I)$. Let z^{opt} and $(\boldsymbol{\beta}^{\text{opt}}, \beta_0^{\text{opt}})$ be the globally optimal value.

$$z^{\text{opt}} := \min_{(\boldsymbol{\beta}, \beta_0)} \| \max\{\mathbf{0}, \boldsymbol{X}\boldsymbol{\beta} + \beta_0 \mathbf{1}\} - \boldsymbol{Y} \|_2^2,$$
$$(\boldsymbol{\beta}^{\text{opt}}, \beta_0^{\text{opt}}) := \arg\min_{(\boldsymbol{\beta}, \beta_0)} \| \max\{\mathbf{0}, \boldsymbol{X}\boldsymbol{\beta} + \beta_0 \mathbf{1}\} - \boldsymbol{Y} \|_2^2.$$

We have that

$$I^{\text{opt}} := \left\{ i \in [m] : \boldsymbol{X}_i^\top \boldsymbol{\beta}^{\text{opt}} + \beta_0^{\text{opt}} > 0 \right\},$$
$$[m] \backslash I^{\text{opt}} := \left\{ i \in [m] : \boldsymbol{X}_i^\top \boldsymbol{\beta}^{\text{opt}} + \beta_0^{\text{opt}} \leq 0 \right\},$$

are the corresponding active, inactive set of $(\boldsymbol{\beta}^{\text{opt}}, \beta_0^{\text{opt}})$, respectively. Hence $z^\sigma(I)$ satisfies:

Proposition 3.2.2. *For any $I \subseteq [m]$, $z^{opt} \leq z^\sigma(I)$. Moreover, there exists an $I \subseteq [m]$ such that $z^{opt} = z^\sigma(I)$.*

Proof of Proposition 3.2.2 can be found in Appendix C.3. Thus, we can use the upper bound $z^\sigma(I)$ instead of $z^*(I)$ to design the algorithm, which we present below.

Insight for Algorithm 4. Recall the definition of the active set, the challenge part for solving ReLU-regression is to determine the optimal active set I^{opt}, i.e., the set of indices $i \in I^{\text{opt}}$,

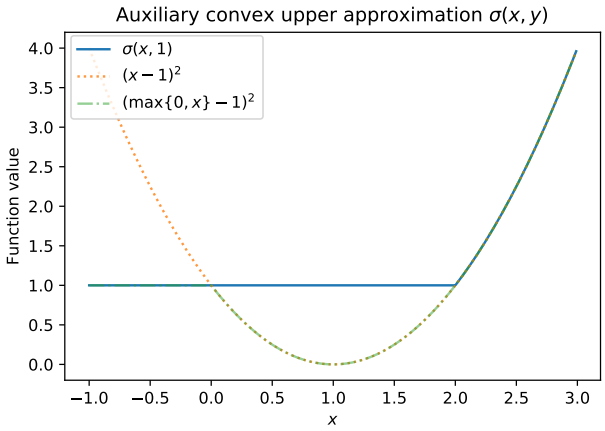


Figure 3.2: Function $\sigma(x, y)$ with $y = 1$.

$\boldsymbol{X}_i^\top \boldsymbol{\beta}^{\text{opt}} + \beta_0^{\text{opt}} > 0$. Since the objective is to minimize the ℓ_2 square loss, then given any feasible solution $(\boldsymbol{\beta}, \beta_0)$, by definition, the i^{th} row will contribute $\left(\max\{0, \boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0\} - Y_i\right)^2$ to the objective value. Suppose that the given $(\boldsymbol{\beta}, \beta_0)$ satisfies $\boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0 \leq 0$. Then, for some $i \in [m]$ such that $0 \ll Y_i$, the i^{th} row contributes a large value to objective. Therefore, we observe that the greater the Y_i, the more likely that the index i belongs to the active set. However, there are usually some "bad indices" with large Y_i but are not in the active set. Thus, in Step 6 and 7 of Algorithm 4, we set a parameter $k \in \mathbb{N}_+$, and enumerate all these possible "bad indices" with cardinality less than or equal to k. More specifically, the first "bad index" i_1 plays a role as a *threshold* that removes the samples $\{(\boldsymbol{X}_\ell, Y_\ell)\}_{\ell=1}^{i_1}$, i.e., the samples with small Y_i; and the rest $i_2, \ldots, i_j$ picks out the "bad indices" with large Y_i but not in the active set. That is to say, we select an *active set* I, and "believe" that each sample $\{(\boldsymbol{X}_i, Y_i)\}_{i \in I}$ in the active set are exactly on the non-zero part of ReLU function, see Figure 3.3. Thus an inactive set with cardinality $i_1 + (k-1)$ is removed from the sample set. This is a key intuition that leads to our approximate algorithm, Algorithm 4, which essentially explores a polynomial number of such *active subsets* with the property that larger the value of Y_i, the more likely the corresponding index i belongs to the set.

Clearly, for each $j = 1, \ldots, k$, there are $\binom{m}{j}$ distinct subsets $\{i_1, \ldots, i_j\}$ in $\{1, \ldots, m\}$. For each picked j indices $i_1, \ldots, i_j$, Algorithm 4 requires to solve a convex optimization

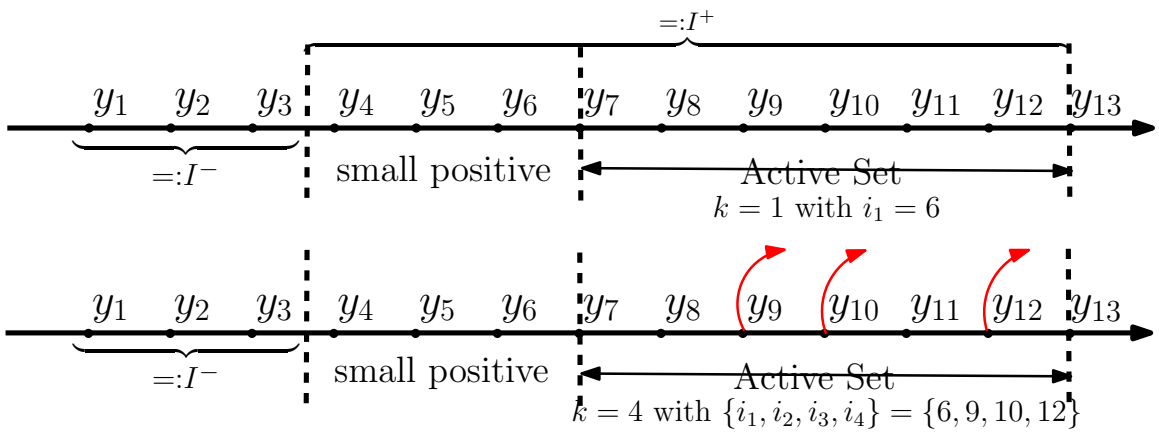


Figure 3.3: Intuition of active set selection.

Algorithm 4 Generalized Approximation Algorithm

Input: A set of n samples $\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n$, a positive-label index set $\mathcal{I}^+ = \{1, \ldots, m\}$ such that $0 < Y_1 \leq Y_2 \leq \cdots \leq Y_m$, a negative-label index set $\mathcal{I}^- = \{m+1, \ldots, n\}$, a fixed integer $k \geq 1$.

Output: A feasible (n/k)-approximate solution $(\boldsymbol{\beta}, \beta_0)$ for the ReLU-regression problem.

1: **for** $j = 1, \ldots, k$ **do**

2: Pick j distinct indices $i_1, \ldots, i_j$ such that $0 \leq i_1 < \ldots < i_j \leq m$.

3: Set the inactive set based on $i_1, \ldots, i_j$ with cardinality $i_1 + (j-1)$ as follows:

$$\begin{cases} \{1, \ldots, i_1\} \cup \left(\bigcup_{\ell=2}^{j} \{i_\ell\}\right) & \text{for } j \geq 2, \\ \{1, \ldots, i_1\} & \text{for } j = 1. \end{cases}$$

4: Set the active set I to be the complement of the inactive set as:

$$I := \left(\bigcup_{\ell=1}^{j-1} \{i_\ell + 1, \ldots, i_{\ell+1} - 1\}\right) \cup \{i_j + 1, \ldots, m\} \subseteq \mathcal{I}^+.$$

5: For each active set, solve the following convex optimization problem:

$$(\boldsymbol{\beta}^I, \beta_0^I) \leftarrow \underset{(\boldsymbol{\beta}, \beta_0)}{\arg\min}\, f_I^\sigma(\boldsymbol{\beta}, \beta_0) + \phi(\boldsymbol{\beta}, \beta_0),$$
$$z^\sigma(I) \leftarrow \min_{(\boldsymbol{\beta}, \beta_0)} f_I^\sigma(\boldsymbol{\beta}, \beta_0) + \phi(\boldsymbol{\beta}, \beta_0).$$

6: Repeat for all possible choices of $i_1, \ldots, i_k$.

7: **end for**

8: **return** $(\hat{\boldsymbol{\beta}}, \hat{\beta}_0)$ which corresponds to the minimum $z^\sigma(I)$ among all the I's examined.

problem, thus the total running time of Algorithm 4 is

$$\left(\sum_{i=1}^{k} \binom{n}{i}\right) T = O\left(n^k T\right),$$

where T is the running time of solving a convex optimization problem

$$(\boldsymbol{\beta}^I, \beta_0^I) \leftarrow \underset{(\boldsymbol{\beta},\beta_0)}{\arg\min}\, f_I^{\sigma}(\boldsymbol{\beta}, \beta_0) + \phi(\boldsymbol{\beta}, \beta_0).$$

Thus, Algorithm 4 is a polynomial-time algorithm.

Theorem 9 (Approximation Ratio)**.** *Algorithm 4 is an* (n/k)*-Approximation Algorithm, i.e., if* z^{approx} *is the objective value of the* $(\hat{\boldsymbol{\beta}}, \hat{\beta}_0)$ *returned from Algorithm 4, and* z^{OPT} *is the globally optimal value of ReLU-regression, then:*

$$z^{\text{OPT}} \leq z^{\text{approx}} \leq \frac{n}{k} z^{\text{OPT}}.$$

Insight for Theorem 9. The proof idea of proving this theorem is to show that, even in the worst-case, the top-k "bad indices" will be partitioned into the correct set, which helps to guarantee the multiplicative approximation ratio. See Appendix D for details.

Comparison with results in the literature:

- Theorem 4.1 in [82] showed that there exists an algorithm to solve the 2-layer ReLU DNNs in time $O(2^w n^{pw}\text{poly}(n, d, w))$ with number of samples n, dimension of input d, maximum width of ReLU network w. We want to point out that the running time grows exponentially in the dimension of input data X_i. In contrast, our algorithm guarantees to find out an approximate solution within polynomial time. The numerical results reported in later sections demonstrate the efficiency and the scalability for high dimensional instances. Although Algorithm 1 in [82] is designed to find global optimality, the computational complexity $O(2^w n^{dw}\text{poly}(n, d, w))$ makes their algorithm intractable for high dimensional instances.

- In [84], Manurangsi and Reichman show that minimizing the squared training error of a one-node neural network is NP-hard to approximate within the factor $(nd)^{1/(\log\log(nd))^{O(1)}}$ (in fact, m samples $\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^m$ in [84]'s setting is equivalent to nm samples in our

setting based their polynomial-time reduction). There is a significant gap between the upper bound from Algorithm 4 and this lower bound. The reason why there exists a significant gap is still an open question. Either there exist some other reductions with greater approximation ratio, or there exists a better polynomial algorithm to solve the ReLU regression problem with a smaller approximation ratio, or both are possible.

An important consequence of Theorem 9 is the following. Below, we say that the ReLU-regression problem is *realizable*, when there exists a true solution $(\boldsymbol{\beta}^*, \beta_0^*)$ with zero objective value.

Corollary 3.2.1 (Realizable case)**.** *When the ReLU-regression problem is realizable, Theorem 9 implies that Algorithm 4 gives a polynomial-time approach that solves the ReLU-regression problem exactly.*

Comparison with results in the literature.

- Kakade et al. [94] proposed the GLM-tron and L-Isotron algorithm to optimize the generalized linear and single index models with isotonic regression. Kakade et al. showed that: the fixed design error (prediction error) obtained from GLM-tron algorithm and L-Isotron is upper bounded by $O(\sqrt{\log(n/\delta)/n})$, $O([\log(n/\delta)/n]^{1/3})$, respectively with constant $\delta \in (0, 1)$ and n the number of samples. In contrast to the thesis, Kakade et al. first assumed the underlying statistical model for input samples, and second required the boundedness of its "activation function" (i.e., u in [94]) within $[0, 1]$.

- In [84], the authors observed that the realizable case can be solved using LP. This observation could also be viewed in Corollary 3.2.1. Since it is sufficient to consider the set of samples with *positive-response* $Y_i > 0$, when the number of positive-response samples is greater than dimension d, the exact solution $\boldsymbol{\beta}^*$ can be obtained by solving the convex problem $\min_{\boldsymbol{\beta}} \sum_{i \in \{i: Y_i > 0\}} (\boldsymbol{X}_i^\top \boldsymbol{\beta} - Y_i)^2$.

- Soltanolkotabi in [83] and Kalan et al. in [85] studied the problem of learning one node ReLU neural network with *i.i.d.* random Gaussian distribution observation samples via gradient descent (GD) method and stochastic gradient descent (SGD) method in the realizable case. Soltanolkotabi showed that the gradient descent, when starting from the origin, converges at a linear rate to the true solution (with additive error) when the number of samples is sufficiently large. Kalan et al. in [85] discussed the stochastic version that mini-batch stochastic gradient descent when suitably initialized, converges at a geometric rate to the true solution (with additive error). In contrast, our Algorithm 4 does not need to assume the data is *i.i.d.* random Gaussian, and the results hold in general. Finally, the SGD method requires a close enough initialization, which is not required by Algorithm 4.

3.2.2 Training ReLU-regression With Underlying Statistical Model

Now we consider the scenario when samples are generated from an underlying statistical model specified as follows. Assume a training sample set $\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n \in \mathbb{R}^d \times \mathbb{R}$ generated from a "true" statistical model as follows:

Definition 3.2.1. ***Statistical Model.*** *Each output Y_i is generated based on the following model,*

$$Y_i = \max\{0, \boldsymbol{X}_i^\top \boldsymbol{\beta}^* + \beta_0^*\} + \epsilon_i, \quad i = 1, \ldots, n,$$

where $\boldsymbol{\beta}^*, \beta_0^*$ are unknown and fixed true parameters, which may be distinct from $\left(\boldsymbol{\beta}^{opt}, \beta_0^{opt}\right)$ as the optimal solution of ReLU-regression. We further assume that $\boldsymbol{\beta}^*, \beta_0^*$ belongs to a convex compact set $\Theta \subseteq \mathbb{R}^p \times \mathbb{R}$. For $i = 1, \ldots, n$, $\boldsymbol{X}_i$, ϵ_i are i.i.d. random variables that are generated from some underlying distributions $\mathcal{N}, \mathcal{D}$, respectively. Finally, we assume the distribution $\mathcal{N}$ satisfies the following properties:

1. $\mathbb{E}_{\boldsymbol{X}\sim\mathcal{N}}[\boldsymbol{X}] = \mathbf{0}_d$, *and* $Var_{\boldsymbol{X}\sim\mathcal{N}}(\boldsymbol{X}) = \boldsymbol{\Sigma}$.

2. *Unique Optimal Property: Let* $\text{Supp}_{\mathcal{N}} \subseteq \mathbb{R}^d$ *be the support of distribution* $\mathcal{N}$*. For any* $(\boldsymbol{\beta}^*, \beta_0^*) \in \Theta$*, there exists* $d+1$ *vectors* $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d, \boldsymbol{v}_{d+1} \in \text{Supp}_{\mathcal{N}}$ *such that*

$$\boldsymbol{v}_i^\top \boldsymbol{\beta}^* + \beta_0^* > 0, \ \forall i = 1, \ldots, d, d+1,$$

and in addition, $(\boldsymbol{v}_1, 1), \ldots, (\boldsymbol{v}_d, 1), (\boldsymbol{v}_{d+1}, 1) \in \mathbb{R}^{d+1}$ *are linearly independent. This property is used to ensure that the global optimal solution is unique and can be identified from ReLU.*

3. $\mathcal{D}$ *is a Gaussian distribution with zero mean and variance* γ^2*.*

To be concise, define Δ^2 *as*

$$\Delta^2 := \textit{Var}_{\boldsymbol{X} \sim \mathcal{N}}(\boldsymbol{X}^\top \boldsymbol{\beta}^* + \beta_0^*) = (\boldsymbol{\beta}^*)^\top \boldsymbol{\Sigma} \boldsymbol{\beta}^*.$$

Connections with Thresholding Methods. Let $S_n^y(\boldsymbol{\beta}, \beta_0)$ be the objective function used in Algorithm 4 with parameter $k = 1$ as follows:

$$S_n^y(\boldsymbol{\beta}, \beta_0) := \frac{1}{n} \cdot \Bigg[\sum_{i \in I(y)} \sigma(\boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0, Y_i) + \sum_{i \in \mathcal{I}^+ \setminus I(y)} (\boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0 - Y_i)^2 + \sum_{i \in \mathcal{I}^-} (\max\{0, \boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0\} - Y_i)^2 \Bigg],$$

where $\mathcal{I}^+ = \{i : Y_i > 0\}$, $\mathcal{I}^- = \{i : Y_i \leq 0\}$, and $I(y) = \{i : 0 < Y_i \leq y\}$ for some $y > 0$. By setting *thresholding parameter* $y = Y_{i_1}$, we have $\{0, 1, \ldots, i_1\}$ corresponds to $I(y)$ and $\{i_1 + 1, \ldots, m\}$ corresponds to $[m] \setminus I(y)$. As we change the thresholding parameter y, we are essentially picking different values of i_i.

To derive the main results in this setting, we follow a few steps. First, using classical results in ([105], p.40) and [106], we obtain

Proposition 3.2.3. *As $n \to \infty$, the objective function*

$$S_n^y(\boldsymbol{\beta}, \beta_0) \to \mathbb{E}_{\boldsymbol{X}\sim\mathcal{N},\epsilon\sim\mathcal{D}}\Big[\psi_y(\boldsymbol{X}^\top\boldsymbol{\beta} + \beta_0, Y)\Big],$$

for almost every sequence $\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n$, where

$$Y_i = \max\{0, \boldsymbol{X}_i^\top\boldsymbol{\beta}^* + \beta_0^*\} + \epsilon_i,$$

and the auxiliary function $\psi_y(\cdot,\cdot)$ is

$$\begin{aligned}
&\psi_y(\boldsymbol{X}^\top\boldsymbol{\beta} + \beta_0, Y) \\
:= &\begin{cases}
\sigma(\boldsymbol{X}^\top\boldsymbol{\beta} + \beta_0, Y) & \text{if } 0 < Y \le y, \\
(\boldsymbol{X}^\top\boldsymbol{\beta} + \beta_0 - Y)^2 & \text{if } y < Y, \\
(\max\{0, \boldsymbol{X}^\top\boldsymbol{\beta} + \beta_0\} - Y)^2 & \text{if } Y \le 0.
\end{cases}
\end{aligned}$$

Proposition 3.2.4. *Assume the statistical model specified above. As $n \to \infty$, the least square estimator $(\beta^{opt}, \beta_0^{opt})$ obtained from solving the ReLU-regression problem is strongly consistent, i.e., they converge to the true parameter $(\boldsymbol{\beta}^*, \beta_0^*)$ almost surely. Moreover, as $n \to \infty$,*

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^n \left(\max\{0, \boldsymbol{X}_i^\top\boldsymbol{\beta} + \beta_0\} - Y_i\right)^2 \\
\to &\, \mathbb{E}_{\boldsymbol{X}\sim\mathcal{N},\epsilon\sim\mathcal{D}}\left[\left(\max\{0, \boldsymbol{X}^\top\boldsymbol{\beta} + \beta_0\} - Y\right)^2\right],
\end{aligned}$$

and

$$\begin{aligned}
&\min_{(\beta,\beta_0)\in\Theta} \mathbb{E}_{\boldsymbol{X}\sim\mathcal{N},\epsilon\sim\mathcal{D}}\left[\left(\max\{0, \boldsymbol{X}^\top\boldsymbol{\beta} + \beta_0\} - Y\right)^2\right] \\
= &\, \mathbb{E}_{\boldsymbol{X}\sim\mathcal{N},\epsilon\sim\mathcal{D}}\left[\left(\max\{0, \boldsymbol{X}^\top\boldsymbol{\beta}^* + \beta_0^*\} - Y\right)^2\right] = \gamma^2.
\end{aligned}$$

Combining Proposition 3.2.3 and Proposition 3.2.4, we obtains the following asymptotic bound for Algorithm 4:

Theorem 10 (Asymptotic Bound)**.** *Assume the statistical model specified above. Let* z^{asy} *be the optimal value of the asymptotic objective function* $\mathbb{E}_{\boldsymbol{X}\sim\mathcal{N},\epsilon\sim\mathcal{D}}\left[\psi_y(\boldsymbol{X}^\top\boldsymbol{\beta}+\beta_0,Y)\right]$ *for all* $y>0$*, i.e.,*

$$z^{\text{asy}}=\min_{y\geq 0}\min_{(\boldsymbol{\beta},\beta_0)\in\Theta}\mathbb{E}_{\boldsymbol{X}\sim\mathcal{N},\epsilon\sim\mathcal{D}}\left[\psi_y(\boldsymbol{X}^\top\boldsymbol{\beta}+\beta_0,Y)\right],$$

then

$$\gamma^2\leq z^{\text{asy}}\leq\frac{3\gamma^2}{2}+\frac{2+2\Delta^2}{\sqrt{2\pi}}\gamma.$$

Insight for Theorem 10. On a high-level, the optimal value of the asymptotic objective function can be represented as a sum of several easy-to-verify conditions. Then we give upper bounds for each of the conditions simultaneously to achieve the final result. Proof is given in Section C.5.

Note that the upper bound for the asymptotic optimal value z^{asy} only depends on the variance Δ^2 and γ^2. Therefore, when the sample set is generated from the underlying statistical model 3.2.1, we have the following corollary:

Corollary 3.2.2 (Asymptotic Approximation Ratio)**.** *Assume the statistical model specified above. As* $n\to\infty$*, the solution obtained from Approximation Algorithm 4 provides an asymptotic multiplicative approximation ratio*

$$\rho\leq\frac{3}{2}+\frac{2+2\Delta^2}{\sqrt{2\pi}}\frac{1}{\gamma},$$

which is independent of the sample size n*. Moreover, this guarantee can be achieved by only computing* $S_n^y(\boldsymbol{\beta},\beta_0)$ *with* $y=0$*.*

We note the following. As the variance of the noise tends to zero, the multiplicative

approximation ratio ρ obtained in Corollary 3.2.2 goes to infinity. However, since the upper bound of z^{asy} is in the order $O(\gamma)$, z^{asy} will also tend to zero.

Comparison with results in the literature: Recently, there is a large number of results that discuss how to use the SGD type algorithms to achieve locally or globally optimal solution efficiently, when there is an underlying statistical model:

- The pioneer work [93] gave a fast, greedy algorithm that can find a fairly good set of parameters quickly based on good initialization using "complementary priors" in a reasonable time. Later, [107] gave empirical evidence that simple two-layer neural networks have good sample expressivity in the over-parameterized case. These earlier works did not provide theoretical guarantees.

- Kalai and Sastry [95] proposed an isotron algorithm that provably learns single index models (SIM) in polynomial time. Comparing with our work, the asymptotic result in this paper does not require the realizable assumption $y_i = u(\boldsymbol{w} \cdot \boldsymbol{x}_i)$ for the idealized SIM problem.

- Oymak et al. [108] focused on minimizing a least-squares objective subject to a constraint defined as the sub-level set of a penalty function and is a related version of the ReLU-regression problem. The authors show the convergence guarantee of the gradient projection algorithm, which can be viewed as a work that gives a non-asymptotic empirical risk. Note that the objective function of the ReLU-regression problem is ℓ_2-norm of ReLU activation instead of the linear function in [108].

- Soltanolkotabi in [83] and Kalan et al. [85] focus on the case with zero noise.

- Kakade et al. [94] provided efficient algorithms for learning the generalizations of linear regression with provable guarantees on the predict error. Compared to ReLU-regression, their guarantees request additional assumptions on the underlying statistical model and the ground truth.

- Brutzkus and Globerson in [88] showed that when there is no noise and when the input is Gaussian distributed, a one-hidden-layer neural network with ReLU activation function can be trained exactly in polynomial time with gradient descent.

- Du et al. in [89] showed that without any specific forms of the input distribution, (1) (stochastic) gradient descent with random initialization can learn the convolutional filter in polynomial time, and (2) its convergence rate depends on the smoothness of the input distribution function. Later, Du et al. in [90] showed that: learning a one-hidden-layer ReLU neural network, (1) with a specific randomized initialization, the gradient descent converges to the ground truth with high probability, (2) the objective function does have a spurious local minimum (i.e., the local minimum plays a non-trivial role in the dynamics of gradient descent). Note that these two papers [90, 89] need a proper initialization to achieve their results.

- Goel et al. in [92] proposed an algorithm – *Convotron*, which captures commonly used schemes from computer vision to learn on-hidden-layer neural networks with a leaky ReLU activate function. The authors show that the convotron algorithm properly recovers the unknown weight vector under some distributional conditions without special/random initialization scheme or tuning of the learning rate. In contrast to our work, their convergence results depend on the "no bias" property of their leaky ReLU function and distributions to be symmetric about the original, which may be restricted in practice.

- Zhang et al. in [99] studied the problem of learning one-hidden-layer neural networks with ReLU activation function, where the inputs are sampled from standard Gaussian distribution and the outputs are generated from a noisy teacher network of width K. The authors show that: gradient descent with tensor initialization can linearly converge to the ground-truth parameters $\boldsymbol{W}^*$ with an additional additive error ϵ, when

the sample size satisfies

$$N \geq \max \left\{ \begin{array}{l} \epsilon^{-2} d \text{poly}(\boldsymbol{W}^*, K), \\ \log(1/\epsilon) d \log d \text{poly}(\boldsymbol{W}^*, K) \end{array} \right\},$$

in Theorem 4.2 (linear convergence) and Lemma 4.5 (tensor initialization) in [99] simultaneously. Moreover, the additive distance statistical error ϵ for parameter $\boldsymbol{W}$ in [99] leads to an additive error for the optimal value of the asymptotic objective function, which cannot be bounded by the multiplicative ratio proposed in Theorem 3 of our paper.

- Laurent et al. [109] studied the loss surface of neural networks equipped with a hinge loss criterion and ReLU or leaky ReLU nonlinearities. Moreover, the authors prove that global minima with zero loss must be trivial, while minima with nonzero loss are necessarily non-differentiable for many fully connected networks. This global minima results can also be viewed in our paper. If the global minima have zero loss, the optimization problem is realizable, which can be solved to global optimality within polynomial time; while global minimization does not equal zero, the ReLU-regression problem is NP-hard to solve.

We also review concurrent papers [110, 100, 111, 112] focusing on non-asymptotic population risk bounds for the completeness of literature survey.

- In [110], Wang et al. present a stochastic gradient descent (SGD) algorithm, which provably trains a one-hidden-layer ReLU neural network to achieve global optimality on the task of binary classification with a hinge loss objective function. In contrast, we focus on the ℓ_2-norm empirical loss for the ReLU-regression problem.

- Cao and Gu [100] proposed a novel algorithm called approximate gradient descent for training CNNs. The authors show that with high probability, the proposed algorithm with random initialization grants a linear convergence to the ground-truth

parameters up to a statistical precision. The authors show that the convergence result holds for monotonic and Lipschitz continuous activation functions. The authors point out that the proposed sample complexity beats existing results and matches the information-theoretic lower bound for learning one-hidden-layer CNNs with linear activation functions. The sample complexity guarantee for this work is better than the sample complexity given from [92], but request additional Gaussian distribution for input samples.

- In [111], Diakonikolas et al. gave a constant-factor approximation algorithm for ReLU assuming the underlying distribution satisfies some weak concentration and anti-concentration conditions, and obtain a polynomial-time approximation scheme for any subgaussian distribution. The authors prove that: when samples are i.i.d. from some isotropic log-concave distribution, for additive error ϵ, sample dimension d, there is an algorithm that uses $\tilde{O}(d/\epsilon^2)$ samples, runs in time $\tilde{O}(d^2/\epsilon^2)$, and achieves population risk $O(\text{opt}) + \epsilon$ with high probability on a convex surrogate for the empirical risk, where opt denotes the optimal population risk for ReLU regression. This work focuses on finding parameter $\boldsymbol{w}$ for population risk with additive approximation guarantee. Since opt $\ll 1$ is assumed for the optimal parameter, then in contrast, the population risk $O(\text{opt}) + \epsilon$ proposed above may lead to non-constant multiplicative approximation ratio.

- In [112], Frei et al. studied the learning problem of a single neuron with gradient descent in the agnostic PAC learning setting. The authors show that: when there is no relationship between labels y and samples x (agnostic learning), the gradient descent achieves $O(\text{opt}) + \epsilon$ population risk in polynomial time; when labels y takes the form $y = \sigma(\boldsymbol{w}^\top x) + \xi$ for zero-mean sub-Gaussian noise ξ (teacher learning), the population risk guarantees for gradient descent improve to opt$+\epsilon$. Similarly, this work gives an additive population risk guarantee for ReLU regression problem. Therefore, since

opt $\ll 1$ is assumed for the optimal parameter, then the population risk $O(\text{opt}) + \epsilon$ proposed above may lead to non-constant multiplicative approximation ratio.

3.3 Numerical results

In this section, we present numerical examples using simulated data to compare our algorithm with three other methods: (1) the sorting method (a simplified version of Algorithm 4 which we describe below in this section), (2) sorting followed by an iterative heuristics, (3) gradient descent methods, (4) sorting followed by gradient descent methods, (5) stochastic gradient descent methods. All numerical experiments are implemented on MacBookPro13 with 2 GHz Intel Core i5 CPU and 8 GB 1867 MHz LPDDR3 Memory. Each optimization step of the sorting method (Algorithm 4) and each optimization step of the iterative method (Algorithm 9 in Appendix C) are solved using Gurobi 7.0.2 in python 3.5.3.

3.3.1 Simulated examples

We perform numerical experiments in the following settings.

1. Given a vector $\boldsymbol{\mu} \in \mathbb{R}^d$, and a positive semidefinite matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$, the true solution $\boldsymbol{\beta}^*$ is generated from the Gaussian distribution $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Specifically, $\boldsymbol{\beta}^*$ in Figure [C.2, C.3, 3.4] are generated from $N(0.5 \cdot \mathbf{0}_d, 10 \cdot \boldsymbol{I}_d)$.

2. Both training set and testing set contain n samples. For each sample $(\boldsymbol{X}_i, Y_i) \in \mathbb{R}^d \times \mathbb{R}$ in training set, the observation sample $\boldsymbol{X}_i = (\boldsymbol{X}_{ij})_{j=1}^d$ is generated by setting $\boldsymbol{X}_{ij} = 1, \boldsymbol{X}_{ij} = -1$ with probability $P/2$ and $\boldsymbol{X}_{ij} = 0$ with probability $1 - P$, independently. In the rest of this sections, we refer to $P = \mathbb{P}(\{\boldsymbol{X}_{ij} = 1\} \cup \{\boldsymbol{X}_{ij} = -1\})$ as the level of *sparsity*. Moreover, in the realizable case we perturbed the data to guarantee that the globally optimal solution is unique. Assume that $\boldsymbol{\beta}_i^* \neq 0$ for all $i \in [d]$. The first d samples $\boldsymbol{X}_i$ are obtained as $\boldsymbol{X}_i \leftarrow \boldsymbol{e}_i \cdot \text{sgn}(\boldsymbol{\beta}_i^*)$ for all $i = 1, \ldots, d$ in the training set, in which $\boldsymbol{e}_i \in \mathbb{R}^d$ is a vector with one on its i^{th} component and

zero otherwise, and $\mathrm{sgn}(x)$ equals to one when $x > 0$ and equals to zero otherwise.

3. Note that the constant term $\boldsymbol{\beta}_0^*$ can be achieved via adding one dimension with value one to each $\boldsymbol{X}_i$. To simplify, we decide to use $\boldsymbol{\beta}_0^* = 0$. The response variable Y_i is thus computed as $Y_i = \max\{0, \boldsymbol{X}_i^\top \boldsymbol{\beta}^*\} + \epsilon_i$ with $\epsilon_i \sim N(0, \rho\sigma)$, where σ and ρ are set in the following way:

   σ: the sample variance σ is computed based on the following procedures:

$$\begin{aligned} Z_i &\leftarrow \boldsymbol{X}_i^\top \boldsymbol{\beta}^*, \ \forall\, i = 1, \ldots, n; \\ \bar{Z} &\leftarrow \tfrac{1}{n} \textstyle\sum_{i=1}^n Z_i; \\ \sigma^2 &\leftarrow \tfrac{1}{n} \textstyle\sum_{i=1}^n (Z_i - \bar{Z})^2. \end{aligned}$$

   ρ: We measure noise level in terms of *signal-to-noise ratio* in decibels (dB). Consider the dB values being $\{6, 10, 20, 30, \infty\}$. The value of Signal-to-Noise (SNR) ratio ρ is given by

$$\text{dB} := 10 \log_{10} \frac{\sigma^2}{\rho^2 \sigma^2} \in \{6, 10, 20, 30, \infty\},$$

   which corresponds to

$$\rho \approx \{0.5, 0.32, 0.1, 0.032, 0\}.$$

4. For each sample $(\tilde{\boldsymbol{X}}_i, \tilde{Y}_i) \in \mathbb{R}^d \times \mathbb{R}$ in the testing set, we generate $\tilde{\boldsymbol{X}}_i, \tilde{Y}_i$ in the same way as the training set.

3.3.2 Algorithms for comparison

In this section, we briefly describe algorithms for comparison in numerical experiments.

Sorting Method (Sorting)

The sorting method is a simplified version of Algorithm 4 with parameter $k = 1$. To reduce the running time, instead of running i_1 for all values ranging from 1 to n, we limit the values of i_1 to be a subset (see Section C.6.1 for details).

Sorting method followed by an Iterative Method (Sorting + Iter)

A natural algorithm which iteratively improve the solution is the following. Fix I and minimize $f^{\sigma}(I)$. Examine the solution and update the choice of I, so that f^{σ} and f^{*} match the current solution. Repeat until a stopping criteria is meet. See Algorithm 9 in Section C.6.2 for details. We use this heuristic to improve the solution obtained from the Sorting method. After obtaining a feasible solution $\hat{\boldsymbol{\beta}}^{\text{sorting}}$, we set the initial point of iterative heuristic to be $\hat{\boldsymbol{\beta}}^{\text{sorting}}$.

Gradient Descent (GD)

The gradient descent method used in numerical experiments is presented in Section C.6.3, see Algorithm 10. Given an initialization $\boldsymbol{\beta}^0 \leftarrow \mathbf{0}_p$, set $\boldsymbol{\beta}^t$ to be the updated solution obtained in $(t-1)^{\text{th}}$ iteration. The gradient $\frac{1}{n}\nabla_{\boldsymbol{\beta}} L(\boldsymbol{\beta}^t)$ used in the t^{th} iteration is given by

$$\frac{1}{n}\sum_{i=1}^{n}(\max\{0, \boldsymbol{X}_i^{\top}\boldsymbol{\beta}^t\} - Y_i)(1 + \text{sgn}(\boldsymbol{X}_i^{\top}\boldsymbol{\beta}^t))\boldsymbol{X}_i,$$

where $L(\boldsymbol{\beta}) = \sum_{i=1}^{n}(\max\{0, \boldsymbol{X}_i^{\top}\boldsymbol{\beta}\} - Y_i)^2$.

Sorting followed by Gradient Descent Method (Sorting + GD)

Similar to the method using sorting followed by gradient descent, here we run sorting algorithms and use the result to initialize GD.

Stochastic Gradient Descent Method (SGD)

Here we initialize $\boldsymbol{\beta}^0$ using zero vector, for both SGD and GD. The only difference between SGD and GD is that: in t^{th} iteration, we uniformly pick a mini-batch B^t of size m from the set of samples $\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n$ at random, and then obtain the gradient used in the t^{th} iteration as

$$\frac{1}{m}\sum_{i\in S^t}(\max\{0, \boldsymbol{X}_i^\top \boldsymbol{\beta}^t\} - Y_i)(1 + \text{sgn}(\boldsymbol{X}_i^\top \boldsymbol{\beta}^t))\boldsymbol{X}_i.$$

See Algorithm 11.

3.3.3 Performance metrics

The solutions $\hat{\boldsymbol{\beta}}$ obtained from the above methods are evaluated in terms of their prediction error, objective value, recovery error, generalization error. The formal definitions are:

- *Prediction Error:*

$$\text{PE} := \sum_{i=1}^n \Big(\max\{0, \boldsymbol{X}_i^\top \hat{\boldsymbol{\beta}}\} - \max\{0, \boldsymbol{X}_i^\top \boldsymbol{\beta}^*\}\Big)^2$$

  where $\{\boldsymbol{X}_i, Y_i\}_{i=1}^n$ is the training sample.

- *Objective Value:* Note that the prediction error defined above is not the objective value obtained by solving the optimization problem. In practice, when $\boldsymbol{\beta}^*$ is unknown, the prediction error cannot be achieved exactly. Thus, we use the objective value (Obj)

$$\text{Obj} := \sum_{i=1}^n \Big(\max\{0, \boldsymbol{X}_i^\top \hat{\boldsymbol{\beta}}\} - Y_i\Big)^2$$

  as an alternative.

- *Recovery Error:* The recovery error measures the distance between the solution $\hat{\boldsymbol{\beta}}$ we obtained and the ground truth $\boldsymbol{\beta}^*$:

$$\text{RE} := \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*\|_2.$$

- *Generalization Error:* The generalization error measures how good the solution $\hat{\boldsymbol{\beta}}$ is when using the objective function with respect to testing set, i.e.,

$$\text{GE} := \sum_{i=1}^{n} \left(\max\{0, \tilde{\boldsymbol{X}}_i^\top \hat{\boldsymbol{\beta}}\} - \tilde{Y}_i\right)^2,$$

  where $\{\tilde{\boldsymbol{X}}_i, \tilde{Y}_i\}_{i=1}^n$ is the testing data.

To compare with the objective function value (which is not divided by n), here the prediction error and generalization error are not divided by the training sample size n.

3.3.4 Notation and parameters

Numerical results are presented in Figure 3.4 of this section, the rest of figures and tables are listed in the Appendix C. Below we present notations and the parameters that used for numerical experiments:

- Each line presented in Figure [C.2, C.3, 3.4] represents the average of the measures or running time obtained from 20 instances under the same settings.
- For the Sorting Method (Algorithm 8), N (the number of splits) used is 10.
- For the Sorting (Algorithm 8) + Iterative Method (Algorithm 9), N (the number of split) is set to 10, and let $\hat{\boldsymbol{\beta}}^{\text{sorting}}$ be the solution obtained from Sorting Method, then the parameters of Iterative Method are set to be:

$$(\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n, \boldsymbol{\beta}^0, T) \leftarrow (\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n, \hat{\boldsymbol{\beta}}^{\text{sorting}}, 20)$$

where $\boldsymbol{\beta}^0$ denotes the starting point, T denotes the maximum number of iterations.

- For the Gradient Descent Method (Algorithm 10), the parameters are set to be

$$(\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n, \boldsymbol{\beta}^0, T, \epsilon, \eta_0, \gamma, \alpha) \leftarrow (\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n, \mathbf{0}_p, 1000, 0.01, 1, 0.03, 0.6)$$

where $\boldsymbol{\beta}^0$ denotes the starting point, T denotes the maximum number of iterations, ϵ is a termination criteria parameter, η_0 denotes the initial stepsize, γ, α are parameters used to adjust step size in each iteration.

- For the Sorting (Algorithm 8) + Gradient Descent Method (Algorithm 10), N (the number of split) is set to be 10, and let $\hat{\boldsymbol{\beta}}^{\text{sorting}}$ is as above, and the parameters of the Gradient Descent Method are set to be:

$$(\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n, \boldsymbol{\beta}^0, T, \epsilon, \eta_0, \gamma, \alpha) \leftarrow (\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n, \hat{\boldsymbol{\beta}}^{\text{sorting}}, 1000, 0.01, 1, 0.03, 0.6).$$

- For the Stochastic Gradient Descent Method (Algorithm 11), parameters are set to be:

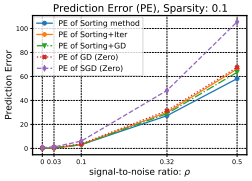
$$(\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n, \boldsymbol{\beta}^0, T, \epsilon, \eta_0, \gamma, \alpha, m) \leftarrow (\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n, \mathbf{0}_p, 1000, 0.01, 1, 0.03, 0.6, \lfloor 0.1n \rfloor).$$

3.3.5 Summary of numerical experiments

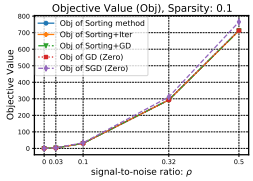
Based on the results reported in Figure [C.2, C.3, 3.4] and Tables in Appendix C.8, some preliminary conclusions can be draw as follows:

- *Prediction Error:* The empirical prediction error compares as

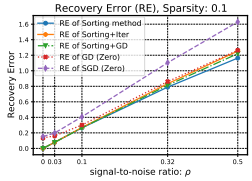
$$\text{PE}^{\text{sorting}} \leq \text{PE}^{\text{sorting + iter}} \approx \text{PE}^{\text{sorting + GD}} \approx \text{PE}^{\text{GD}} \leq \text{PE}^{\text{SGD}}$$



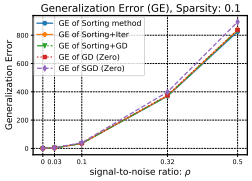
(a) Prediction error



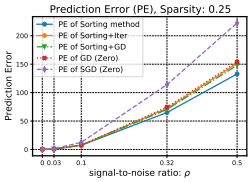
(b) Objective value



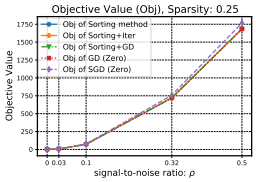
(c) Recovery error



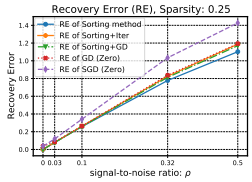
(d) Generalization error



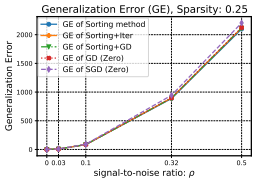
(e) Prediction error



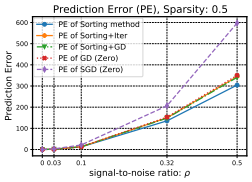
(f) Objective value



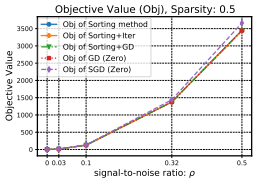
(g) Recovery error



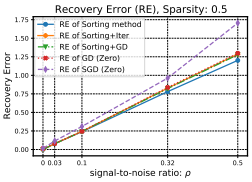
(h) Generalization error



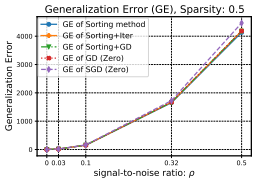
(i) Prediction error



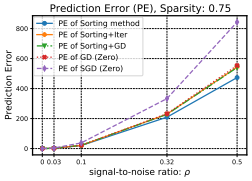
(j) Objective value



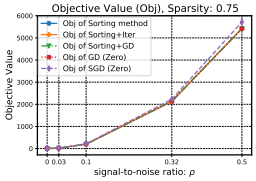
(k) Recovery error



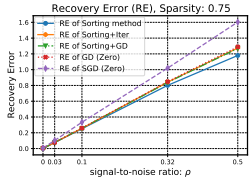
(l) Generalization error



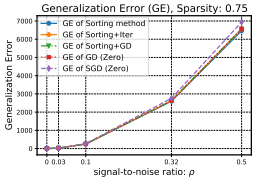
(m) Prediction error



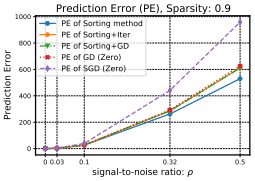
(n) Objective value



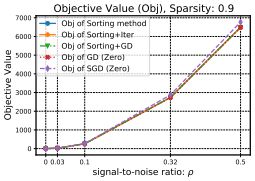
(o) Recovery error



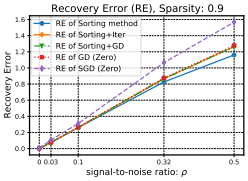
(p) Generalization error



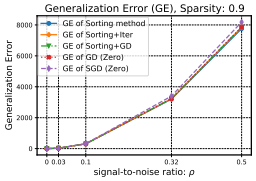
(q) Prediction error



(r) Objective value



(s) Recovery error



(t) Generalization error

Figure 3.4: Numerical Results of sample size $(d, n) = (50, 1000)$ and $\boldsymbol{\beta}^* \sim N(0.5 \cdot \mathbf{1}_p, 50 \cdot \boldsymbol{I}_p)$ with sparsity $\{0.1, 0.25, 0.5, 0.75, 0.9\}$.

where the differences between $\text{PE}^{\text{sorting + iter}}, \text{PE}^{\text{sorting + GD}}, \text{PE}^{\text{GD}}$ are relative smaller than the differences between $\text{PE}^{\text{sorting}}, \text{PE}^{\text{sorting + iter}}$ and $\text{PE}^{\text{GD}}, \text{PE}^{\text{SGD}}$. These empirical results show that the when the output samples $\{Y_i\}$ follows the correct underlying model (which may not be for some real applications), the sorting method performs well in practice.

- *Objective Value:* In most of the cases, objective value satisfies

$$\text{Obj}^{\text{sorting + iter}} \approx \text{Obj}^{\text{sorting + GD}} \approx \text{Obj}^{\text{sorting}} \approx \text{Obj}^{\text{GD}} \leq \text{Obj}^{\text{SGD}}.$$

  The difference between the SGD method and the GD method is large in general since SGD cannot always find out the local minimum solution in a reasonable time. The gaps between the GD method and the rest three methods (Sorting, Sorting + GD, Sorting + Iter) are relatively larger than the differences within the rest three methods. The objective value of the sorting method, when the standard deviation of noise grows, increases most. The sorting + iterative method and sorting + gradient descent method performs almost the same for objective value, which implies that: (1) using iterative method after the sorting really benefits the optimization (comparing with sorting method with smaller objective value); (2) initializing with $\hat{\beta}^{\text{sorting}}$ will improve the performances of GD.

- *Recovery Error:* When the noise variance is small, the recovery error satisfies that

$$\text{RE}^{\text{sorting}} \leq \text{RE}^{\text{sorting + iter}} \approx \text{RE}^{\text{sorting + GD}} \approx \text{RE}^{\text{GD}} \leq \text{RE}^{\text{SGD}}.$$

  As the standard deviation of noise increases, the recovery error obtained from the gradient descent method will not increases as much as the rest three types of methods and finally becomes the best at the point with $\rho = 0.32$.

- *Generalization Error:* The performances of generalization error are very similar to the performances of prediction error. Hence the sorting + iterative method has the strongest generalization power.

- *Running Time:* Empirically, the running time of the sorting method, sorting + iter method, sorting + GD method, and the SGD method satisfies the following:

$$T^{\text{SGD}} \leq T^{\text{sorting}} \leq T^{\text{sorting + iter}} \approx T^{\text{sorting + GD}}$$

  in most of cases. One possible result of the least running time of the SGD method is that SGD cannot find out the local minimum and stops early with fewer iterations. For the GD method with zero initialization, as the size of instances increases, the running time increases faster than the rest four methods. Moreover, the sparsity level, in empirical, has a significant impact on the running time of the GD method.

- *Overfitting:* In simulation results, the sorting+iter algorithm has the lowest objective value and recovery error, whereas the sorting algorithm has the lowest prediction error. We believe that the sorting+iter algorithm is overfitting in some of the cases, while the sorting algorithm is not.

3.4 Conclusions and discussions

After showing that that ReLU-regression is NP-hard, we presented a polynomial-time approximation algorithm for this problem. We showed that for arbitrary data, our algorithm gives a multiplicative guarantee of (n/k) where n is the number of samples, and k is a fixed integer. An important consequence of this result is that in the realizable case, ReLU-regression can be solved in polynomial time. Under a statistical model for training sample, where the data comes from the output of a single node with Relu function with the output being perturbed with Gaussian noise, we can show that the algorithm guarantees are independent of n. To the best of our knowledge, these are the best theoretical performance

guarantees for the solving ReLU-regression, especially in the realizable case and in the case of the statistical data model.

We performed extensive numerical experiments and showed that, in particular, initializing SGD with the output of our approximation algorithm can improve performance in prediction and recovery error, especially when the signal-to-noise ratio is high based on Figure 3.4. In our opinion, this is a crucial empirical observation in the following sense. There is value in coming up with specialized approximation algorithms for various non-convex problems, for which we intend to use gradient descent. The reason is that such approximate algorithms, with theoretical guarantees, provide a good warm-start for SGD, usually a requirement for the SGD algorithm to work well in predict and recovery error for ReLU-regression.

Moreover, we do not want to claim that the sorting algorithm performs better than the SGD-type algorithms (like SVRG or SAGA). Instead, we would like to point out that the solution obtained from the sorting algorithm can be viewed as an initial point with theoretical guarantees even in the model-free case, compared to some other widely-used initialization technique (e.g., method of moments).

Many open questions remain. In the case of the arbitrary training sample model, there is a big gap between the multiplicative guarantee of (n/k) and known lower bound of $(nd)^{1/(\log\log(nd))^{O(1)}}$. In the statistical model, we conjecture that our approximate algorithm is optimal, i.e., performance guarantees cannot be improved. Proving or disproving this conjecture is essential. Another important direction of research is to extend these results to multi-node networks.

Appendices

APPENDIX A

APPENDICES FOR CHAPTER 1

A.1 SDP relaxation

The SPCA problem is equivalent to a nonconvex problem:

$$\begin{array}{ll} \max & \boldsymbol{v}^\top \boldsymbol{A}\boldsymbol{v} \\ \text{s.t.} & \|\boldsymbol{v}\|_2 = 1, \|\boldsymbol{v}\|_0 \le k \end{array} \quad \Leftrightarrow \quad \begin{array}{ll} \max & \text{tr}(\boldsymbol{A}\boldsymbol{V}) \\ \text{s.t.} & \text{tr}(\boldsymbol{V}) = 1, \|\boldsymbol{V}\|_0 \le k^2, \boldsymbol{V} \succeq 0, \text{rank}(\boldsymbol{V}) = 1 \end{array}$$

Further relaxing this by replacing its rank and cardinality constraints with $\mathbf{1}^\top|\boldsymbol{V}|\mathbf{1} \le k$ gives the standard SDP relaxation:

$$\begin{array}{ll} \max & \text{tr}(\boldsymbol{A}\boldsymbol{V}) \\ \text{s.t.} & \text{tr}(\boldsymbol{A}\boldsymbol{V}) = 1, \mathbf{1}^\top|\boldsymbol{V}|\mathbf{1} \le k, \boldsymbol{V} \succeq 0. \end{array} \tag{SDP}$$

A.2 Proof of Proposition 1.2.1

Proof. **Proof of Proposition 1.2.1:** Let $\boldsymbol{v}^* = (\boldsymbol{v}_i^*)_{i=1}^d$ be an optimal solution of SPCA. Then set

$$\begin{cases} g_i^* & \leftarrow \ (\boldsymbol{v}^*)^\top \boldsymbol{w}_i, & i \in [d], \\ \left((\eta_i^j)^*\right)_{j=-N}^N & \leftarrow \ \left((\eta_i^j)^*\right)_{j=-N}^N \in \text{SOS-2 and } \sum_{j=-N}^N \gamma_i^j (\eta_i^j)^* = g_i^*, & i \in I^+, \\ \xi_i^* & \leftarrow \ \sum_{j=-N}^N (\gamma_i^j)^2 \eta_i^j, & i \in I^+, \\ s^* & \leftarrow \ \sum_{i \in I^-} -(\lambda_i - \lambda) g_i^*. \end{cases}$$

We claim that the above solution $(\boldsymbol{v}^*, g^*, \xi^*, \eta^*, s^*)$ is a feasible solution for (Convex-IP) due to the following two parts. First, note that the above setting directly satisfy all the constraints in (Convex-IP) except the constraint $\sum_{i \in I^+} \xi_i + \sum_{i \in I^-} g_i^2 \le 1 + \frac{1}{4N^2} \sum_{i \in I^+} \theta_i^2$.

Second, for the exception constraint, based on the size of the discretization and the structure of SOS-2 constraints, we have $\xi_i^* \leq (g_i^*)^2 + \frac{1}{4N^2}\theta_i^2$ for $i \in I^+$ which implies that exception constraint also holds.

Moreover, the objective value of feasible solution $(\boldsymbol{v}^*, g^*, \xi^*, \eta^*, s^*)$ is

$$\begin{aligned}\lambda_{\text{TH}} + \sum_{i\in I^+}(\lambda_i - \lambda_{\text{TH}})\xi_i^* - s^* \geq & \lambda_{\text{TH}} + \sum_{i\in I^+}(\lambda_i - \lambda_{\text{TH}})(g_i^*)^2 - s^* \\ = & \lambda + \sum_{i\in I^+}(\lambda_i - \lambda_{\text{TH}})((\boldsymbol{v}^*)^\top \boldsymbol{w}_i)^2 + \sum_{i\in I^-}(\lambda_i - \lambda)((\boldsymbol{v}^*)^\top \boldsymbol{w}_i)^2 \\ = & \lambda_{\text{TH}} + \sum_{i=1}^{d}(\lambda_i - \lambda_{\text{TH}})((\boldsymbol{v}^*)^\top \boldsymbol{w}_i)^2.\end{aligned}$$

Note that the optimal solution $\boldsymbol{v}^*$ of SPCA has property $\|\boldsymbol{v}^*\|_2 = 1$ and $\sum_{i=1}^{d} \boldsymbol{w}_i \boldsymbol{w}_i^\top = \boldsymbol{I}_d$. Then $\lambda_{\text{TH}} + \sum_{i=1}^{d}(\lambda_i - \lambda_{\text{TH}})((\boldsymbol{v}^*)^\top \boldsymbol{w}_i)^2 = (\boldsymbol{v}^*)^\top \boldsymbol{A}\boldsymbol{v}^* = \lambda^k(\boldsymbol{A})$. Therefore, $\text{opt}_{\text{convex-IP}} \geq \lambda^k(\boldsymbol{A})$. □

A.3 Proof of Proposition 1.2.2

Proof. **Proof of Proposition 1.2.2:** Let $(\bar{\boldsymbol{v}}, \bar{g}, \bar{\xi}, \bar{\eta}, \bar{s})$ be an optimal solution for Convex-IP. Its optimal value then satisfies the following:

$$\begin{aligned}\text{opt}_{\text{convex-IP}} &= \lambda_{\text{TH}} + \sum_{i\in I^+}(\lambda_i - \lambda_{\text{TH}})\bar{\xi}_i - \bar{s} \\ &= \lambda_{\text{TH}} + \sum_{i\in I^+}(\lambda_i - \lambda_{\text{TH}})\left(\bar{\xi}_i - \bar{g}_i^2 + \bar{g}_i^2\right) - \bar{s} \\ &= \lambda_{\text{TH}} + \sum_{i\in I^+}(\lambda_i - \lambda_{\text{TH}})\left(\bar{\xi}_i - \bar{g}_i^2\right) + \sum_{i\in I^+}(\lambda_i - \lambda_{\text{TH}})\bar{g}_i^2 - \bar{s}.\end{aligned}$$

Since variable s satisfies $\sum_{i\in I^-} -(\lambda_i - \lambda_{\text{TH}})g_i^2 \leq s$, to maximize the objective function, $\bar{s}$ should be equivalent to $\sum_{i\in I^-} -(\lambda_i - \lambda_{\text{TH}})\bar{g}_i^2$, then the above formula can be represented

as

$$
\begin{aligned}
&\lambda_{\mathrm{TH}} + \sum_{i\in I^+}(\lambda_i - \lambda_{\mathrm{TH}})\left(\bar{\xi}_i - \bar{g}_i^2\right) + \sum_{i\in I^+}(\lambda_i - \lambda_{\mathrm{TH}})\bar{g}_i^2 - \bar{s} \\
&= \lambda_{\mathrm{TH}} + \sum_{i\in I^+}(\lambda_i - \lambda_{\mathrm{TH}})\left(\bar{\xi}_i - \bar{g}_i^2\right) + \sum_{i\in I^+}(\lambda_i - \lambda_{\mathrm{TH}})\bar{g}_i^2 + \sum_{i\in I^-}(\lambda - \lambda_{\mathrm{TH}})\bar{g}_i^2 \\
&= \sum_{i\in I^+}(\lambda_i - \lambda_{\mathrm{TH}})\left(\bar{\xi}_i - \bar{g}_i^2\right) + \left(\lambda_{\mathrm{TH}} + \sum_{i=1}^{d}(\lambda_i - \lambda_{\mathrm{TH}})\bar{g}_i^2\right). \qquad \text{(A.1)}
\end{aligned}
$$

By previous results, $\lambda_{\mathrm{TH}} + \sum_{i=1}^{n}(\lambda_i - \lambda_{\mathrm{TH}})\bar{g}_i^2 = \bar{\boldsymbol{v}}^\top \boldsymbol{A}\bar{\boldsymbol{v}}$. Note that due to the ℓ_2-norm constraint $\|\boldsymbol{v}\|_2 \le 1$ and the ℓ_1-norm constraint present in (Convex-IP) problem, we have $\bar{\boldsymbol{v}} \in T_k = \{\boldsymbol{v} \in \mathbb{R}^d : \|\boldsymbol{v}\|_2 \le 1, \|\boldsymbol{v}\|_1 \le \sqrt{k}\} \subseteq \rho \cdot \mathrm{Conv}\,(S_k)$. Therefore $\bar{\boldsymbol{v}}^\top \boldsymbol{A}\bar{\boldsymbol{v}}$ is upper bounded by the value $\rho^2 \cdot \lambda^k(\boldsymbol{A})$.

To upper bound the first term in (A.1), since $g_i = \sum_{j=-N}^{N}\gamma_i^j\eta_i^j,\ \xi_i = \sum_{j=-N}^{N}(\gamma_i^j)^2\eta_i^j$ for $i \in I^+$ and the SOS-2 construction enforces that there are at most two *active* continuous SOS-2 variables η_i^j, η_i^{j+1} such that $\eta_i^j + \eta_i^{j+1} = 1$ with $\eta_i^j, \eta_i^{j+1} \ge 0$ and the other SOS-2 variables are all zeros, then

$$
\begin{aligned}
\xi_i - g_i^2 \ &= \sum_{j=-N}^{N}(\gamma_i^j)^2\eta_i^j - \left(\sum_{j=-N}^{N}\gamma_i^j\eta_i^j\right)^2 \\
&= (\gamma_i^j)^2\eta_i^j + (\gamma_i^{j+1})^2\eta_i^{j+1} - \left(\gamma_i^j\eta_i^j + \gamma_i^{j+1}\eta_i^{j+1}\right)^2 && \text{for } \eta_i^j, \eta_i^{j+1} \text{ active} \\
&= (\gamma_i^{j+1} - \gamma_i^j)^2\eta_i^j(1 - \eta_i^j) && \text{via } \eta_i^j + \eta_i^{j+1} = 1 \\
&\le \max_{j=-N,\ldots,N-1}(\gamma_i^{j+1} - \gamma_i^j)^2 \cdot \frac{1}{4}
\end{aligned}
$$

where in all possible partition of $[-\theta_i, \theta_i]$, the evenly partition of $[-\theta_i, \theta_i]$ achieves the minimum value of $\max_{j=-N,\ldots,N-1}(\gamma_i^{j+1} - \gamma_i^j)^2 = \frac{\theta_i^2}{N^2}$. Hence (A.1) can be upper bounded

as follows:

$$\begin{aligned}\text{opt}_{\text{convex-IP}} &= \sum_{i\in I^+}(\lambda_i - \lambda_{\text{TH}})\left(\bar{\xi}_i - \bar{g}_i^2\right) + \left(\lambda_{\text{TH}} + \sum_{i=1}^{d}(\lambda_i - \lambda_{\text{TH}})\bar{g}_i^2\right) \\ &\leq \frac{1}{4N^2}\sum_{i\in I^+}(\lambda_i - \lambda_{\text{TH}})\theta_i^2 + \rho^2 \cdot \lambda^k(\boldsymbol{A}).\end{aligned}$$

□

A.4 Proof of Proposition 1.2.3

Proof. **Proof of Proposition 1.2.3:** Given the threshold λ_{TH}, the number of splitting points N, the size of set $I_{\text{pos}} = |\{i : \lambda_i > \lambda_{\text{TH}}\}|$, for each $i \in \{i : \lambda_i > \lambda_{\text{TH}}\}$, there are at most $2N$ possible choices of *active* SOS-2 variables, i.e.,

$$\eta_i^j, \eta_i^{j+1} > 0, \text{ for } j = -N, \ldots, 0, \ldots, N-1.$$

Thus there are at most $(2N)^{|I_{\text{pos}}|}$ choices of *active* SOS-2 variables for a Convex-IP problem. For a fixed value of *active* SOS-2 variables, the Convex-IP problem reduces to be a continuous convex optimization problem which can be solved exactly within polynomial time, say T. Thus the Convex-IP can be solved within $(2N)^{|I_{\text{pos}}|} \cdot T$. □

A.5 Proof of Proposition 1.2.4

Proof. **Proof of Proposition 1.2.4:** Based on Proposition 1.2.2, we have

$$\text{opt}_{\text{Pert-Convex-IP}} \leq \rho^2\lambda^k(\bar{\boldsymbol{A}}) + \frac{1}{4N^2}\sum_{i\in I^+}(\lambda_i - \lambda_{\text{TH}})\theta_i^2.$$

Note that $\bar{\boldsymbol{A}} - \boldsymbol{A} = \sum_{i \in I^-} (\bar{\lambda} - \lambda_i) \boldsymbol{w}_i \boldsymbol{w}_i^\top$. Therefore,

$$\begin{aligned}
\rho^2 \lambda^k(\bar{\boldsymbol{A}}) &= \rho^2 \lambda^k \left(\boldsymbol{A} + (\bar{\boldsymbol{A}} - \boldsymbol{A}) \right) \\
&\leq \rho^2 \lambda^k(\boldsymbol{A}) + \rho^2 \lambda^k(\bar{\boldsymbol{A}} - \boldsymbol{A}) \\
&\leq \rho^2 \lambda^k(\boldsymbol{A}) + \rho^2 (\bar{\lambda} - \lambda_{\min}(\boldsymbol{A})).
\end{aligned}$$

□

A.6 Convex-IP Method and Pert-Convex-IP Method

Algorithm 5 presents all the details of the convex IP solved. Algorithm 6 presents all the details of the Pert-Convex-IP solved.

Algorithm 5 Convex-IP Method

Input: Sample covariance matrix $\boldsymbol{A}$, sparse parameter k, size of set I_{pos}, splitting parameter N.
Output: Lower and upper bound of SPCA or ℓ_1-relax based on the choice of θ.
function CONVEX-IP METHOD($\boldsymbol{A}, k, I_{\text{pos}}, N$)
    Set lower bound and warm starting point

$$(\text{LB}, \bar{\boldsymbol{v}}) \leftarrow \text{HEURISTIC METHOD}(\boldsymbol{A}, k, \boldsymbol{v}^0).$$

    Set parameter $\lambda_{I_{\text{pos}+1}} \leq \lambda_{\text{TH}} \leq \text{LB}$ if possible, otherwise set $\lambda_{\text{TH}} \leftarrow \text{LB}$.
    Set splitting points γ_i^j as above based on N and the choice of θ, see Section 1.2.2 [1.2.2] .
    To warm start, add additional splitting points based on the point $\bar{\boldsymbol{v}}$.
    Add cutting-plane (1.2) to the model based on the choice of θ.
    Run Convex-IP problem.
    Set UB $\leftarrow$ Convex-IP if running to the optimal, or the current dual bound obtained from Convex-IP.
    return LB, UB.
end function

Algorithm 6 Pert-Convex-IP Method

Input: Sample covariance matrix $\boldsymbol{A}$, sparse parameter k, size of set I_{pos}, splitting parameter N, maximum number of iterations iter.
Output: Lower and upper bound of SPCA or ℓ_1-relax based on the choice of θ_i.
function PERT-CONVEX-IP METHOD($\boldsymbol{A}, k, I_{\text{pos}}, N$, iter)
    Set lower bound and warm starting point $(\text{LB}, \bar{\boldsymbol{v}}) \leftarrow$ HEURISTIC METHOD($\boldsymbol{A}, k, \boldsymbol{v}^0$).
    Set parameter $\lambda_{I_{\text{pos}+1}} \leq \lambda_{\text{TH}} \leq \text{LB}$ if possible, otherwise set $\lambda \leftarrow \text{LB}$.
    Set parameter $\bar{\lambda} := \max\{\lambda_i : \lambda_i \leq \lambda_{\text{TH}}\} < \lambda_{\text{TH}}$ if possible.
    Set splitting points γ_i^j as above based on N and the choice of θ_i, see Section 1.2.2 [1.2.2].
    To warm start, add additional splitting points based on the point $\bar{\boldsymbol{v}}$.
    while current iteration does not exceed the maximum number of iterations iter or time limit is not up **do**
        Run Pert-Convex-IP problem.
        Set UB $\leftarrow$ Pert-Convex-IP if running to the optimal, or the current dual bound obtained from Pert-Convex-IP.
        Set $\hat{\boldsymbol{v}} \leftarrow$ current feasible solution obtained from Pert-Convex-IP.
        Add additional splitting points based on solution obtained in solving Pert-Convex-IP problem.
        Add cutting-plane (1.2) to the model based on the choice of θ_i.
    end while
    return LB, UB.
end function

A.7 Description of Data Sets

A.7.1 Artificial Data Sets

We first conduct numerical experiments on three types of artificial data sets, denoted as the spiked covariance recovery from the paper [16], the synthetic example from the paper [4], and the controlling sparsity case from the paper [21]. A description of each of these three types of instances is presented below:

Spiked covariance recovery

Consider any covariance matrix $\boldsymbol{\Sigma}$, which has two sparse eigenvectors with dominated eigenvalues and the rest eigenvector are unconstrained with small eigenvalues. Let the first

two dominant eigenvectors $\boldsymbol{v}_1, \boldsymbol{v}_2$ of $\boldsymbol{\Sigma}$ be:

$$[\boldsymbol{v}_1]_i = \begin{cases} \frac{1}{\sqrt{10}} & i = 1, \ldots, 10, \\ 0 & \text{otherwise} \end{cases}, \qquad [\boldsymbol{v}_2]_i = \begin{cases} \frac{1}{\sqrt{10}} & i = 11, \ldots, 20, \\ 0 & \text{otherwise} \end{cases}, \tag{A.2}$$

with the eigenvalues corresponding to the first two dominant eigenvectors be $\lambda_1 \gg 1$ and $\lambda_2 \gg 1$, and the remaining eigenvalues be 1. For example, in our numerical experiments, set $\boldsymbol{\Sigma} \leftarrow 399 \cdot \boldsymbol{v}_1\boldsymbol{v}_1^\top + 299 \cdot \boldsymbol{v}_2\boldsymbol{v}_2^\top + \boldsymbol{I}$.

We have four distinct settings under the spiked covariance recovery case. Let d be the number of features, i.e., the size of the sample covariance matrix of our numerical cases. Let M be the number of samples we generated. We set $d = \{200, 300, 400, 500, 1000\}$ and $M = \{50\}$. Therefore, under each setting of d, we generate M random samples $\boldsymbol{x}_n \sim N(0, \boldsymbol{\Sigma})$, and get our sample covariance matrix $\hat{\boldsymbol{\Sigma}} = \frac{1}{50}\sum_{n=1}^{50} \boldsymbol{x}_n\boldsymbol{x}_n^\top$. In Table 1.4, for each setting, we repeat the experiment for 2 times (case 1, case 2), and compare the dual bounds obtained from all three methods.

Synthetic Example

Given d, let $d_1, d_2, d_3 \in \left\{\lceil \frac{d}{3} \rceil, \lfloor \frac{d}{3} \rfloor\right\}$ such that $d_1 + d_2 + d_3 = d$. Let $\mathbf{0}_{p\times q}$ be the matrix of all zeros with size $p \times q$. Let $\mathbf{1}_p$ be the vector of all ones with length p. Then:

$$\boldsymbol{\Sigma} = \begin{pmatrix} 290 \cdot \mathbf{1}_{d_1}\mathbf{1}_{d_1}^\top + \boldsymbol{I}_{d_1} & \mathbf{0}_{d_1\times d_2} & -87 \cdot \mathbf{1}_{d_1}\mathbf{1}_{d_3}^\top \\ \mathbf{0}_{d_2\times d_1} & 300 \cdot \mathbf{1}_{d_2}\mathbf{1}_{d_2}^\top + \boldsymbol{I}_{d_2} & 277.5 \cdot \mathbf{1}_{d_2}\mathbf{1}_{d_3}^\top \\ -87 \cdot \mathbf{1}_{d_3}\mathbf{1}_{d_1}^\top & 277.5 \cdot \mathbf{1}_{d_3}\mathbf{1}_{d_2}^\top & 582.7875 \cdot \mathbf{1}_{d_3}\mathbf{1}_{d_3}^\top + \boldsymbol{I}_{d_3} \end{pmatrix}. \tag{A.3}$$

In our experiments, we set $d = \{200, 300, 400, 500, 1000\}$, and generate $M = 50$ samples such that $\boldsymbol{x}_n \sim N(0, \boldsymbol{\Sigma})$. Again, the sample empirical covariance matrix is $\hat{\boldsymbol{\Sigma}} = \frac{1}{50}\sum_{n=1}^{50} \boldsymbol{x}_n\boldsymbol{x}_n^\top$. In Table 1.6, for each setting of d, we generate two instances (case 1, case 2) for numerical experiments, and compare dual bounds obtained from all three methods.

Controlling Sparsity

Like the spiked covariance recovery case, the covariance matrix $\mathbf{\Sigma}$ of controlling sparsity case can also be represented as the summation of a term generated by sparse eigenvector with dominated eigenvalue and the remaining part with small eigenvalues. Generate a $d \times d$ matrix $\boldsymbol{U}$ with uniformly distributed coefficients in $[0, 1]$ which can be seen as white noise. Let $\boldsymbol{v} \in \{0, 1\}^d$ be a sparse vector with $\|\boldsymbol{v}\|_0 \leq k$. We then form a test matrix $\mathbf{\Sigma} = \boldsymbol{U}^\top \boldsymbol{U} + \sigma \boldsymbol{v}\boldsymbol{v}^\top$, where σ is the signal-to-noise ratio and is set to 15. In our experiments, we set $d = \{200, 300, 400, 500, 1000\}$ and generate $M = 50$ samples $\boldsymbol{x}_n \sim N(0, \mathbf{\Sigma})$ for $n = 1, \ldots, 50$. Therefore the sample empirical covariance matrix is $\hat{\mathbf{\Sigma}} = \frac{1}{50} \sum_{n=1}^{50} \boldsymbol{x}_n \boldsymbol{x}_n^\top$. In Table 1.8, for each setting of d, we repeat the experiment twice (case 1, case 2), and compare dual bounds obtained from all three methods.

A.7.2 Real Data Sets

We conduct numerical experiments on three types of real data sets, the benchmark pitprops data from [113], biological data from [114, 16, 11] and large-scale data collected from internet.

Pitprops Data

The PitProps data set in [113] (consisting of 180 observations with 13 measured variables) has been a standard benchmark to evaluate algorithms for sparse PCA.

Based on previous work, we also consider the first six $k-$sparse principal components. Note the i-th $k-$sparse principal component $\boldsymbol{v}^i$ is obtained by solving

$$\underset{\|\boldsymbol{v}\|_2=1, \|\boldsymbol{v}\|_0 \leq k}{\arg\max} \quad \boldsymbol{v}^\top \boldsymbol{A}^i \boldsymbol{v}$$

where $\boldsymbol{A}^1 \leftarrow \boldsymbol{A}$ and $\boldsymbol{A}^i \leftarrow (\boldsymbol{I} - \boldsymbol{v}^{i-1}(\boldsymbol{v}^{i-1})^\top)\boldsymbol{A}^{i-1}(I - \boldsymbol{v}^{i-1}(\boldsymbol{v}^{i-1})^\top)$ for $i = 2, \ldots, 6$. Table 1.10 lists the six extracted sparse principal direction with sparse parameter k be

$5-2-2-1-1-1$.

Biological Data

In Table 1.11 we present numerical experiments on four biological data sets. The first two biological data sets (Eisen-1, Eisen-2) are from [11]. The Colon cancer data set is from Alon et al. (1999). The Lymphoma data set is from Alizadeh et al. (2000).

Large-scale Internet Data

In Table 1.11 we also present numerical experiments on internet dataset. This dataset is constructed out of textual posts shared on the popular social media Reddit. Based on prior work [115, 116], the archive of all public Reddit posts shared on Google's Big Query was utilized to obtain a set of 3292 posts from the subreddit r/stress from December 2010 to January 2017. The r/stress community allows individuals to self-report and disclose their stressful experiences and is a support community. For example, two (paraphrased) post excerpts say: "Feel like I am burning out (again...) Help: what do I do?"; and "How do I calm down when I get triggered?". The community is also heavily moderated; hence these 3292 posts were considered to be indicative of actual stress. [116].

Then on this collected set of posts, standard text-based feature extraction techniques were applied per post, starting with cleaning the data (stopword elimination, removal of noisy words, stemming), and then building a language model with the n-grams in a post (n=2). The outcomes of this language model provided us with 1950 features, after including only the top most statistically significant features. Additionally, the psycholinguistic lexicon Linguistic Inquiry and Word Count (LIWC) [117] was leveraged to obtain features aligning with 50 different empirically validated psychological categories, such as positive affect, negative affect, cognition, and function words. These features have been extensively validated in prior work to be indicative of stress and similar psychological constructs [118]. Our final dataset matrix comprised 3092 rows, corresponding to the 3092 posts, and 2000

features in all.

The purpose of testing the sparse PCA technique on this dataset is to identify those features that are theoretically guaranteed to be the most salient in describing the nature of stress expressed in a post. In turn, these salient features could be utilized by a variety of stakeholders like clinical psychologists, and community moderators and managers to gain insights into stress-related phenomenon as well as to direct interventions as appropriate.

The final $\boldsymbol{A}$ matrix can be found on the website:
https://www2.isye.gatech.edu/ sdey30/publications.html

A.8 Comparison with Existing Primal Heuristics for Lower Bounds

In this section, we compare our method Algorithm 2 for obtaining good primal feasible solutions with two standard heuristics methods for sparse PCA in the literature: truncated power method (TPM, [38]), generalized power method (GPM, [12]) with ℓ_0-penalty. See Table A.1 for a comparison on all the real instances. As we can see, all the methods produce

Table A.1: Compare with existing primal methods

Instance	SPCA-Primal (Our method)		TPM		GPM	
	LB	Time	LB	Time	LB	Time
Pitprops $k = 5$	**3.406**	0.1	**3.406**	0.0	**3.406**	0.1
Eisen-1 $k = 10$	**17.335**	0.0	**17.335**	0.0	**17.335**	2.3
Eisen-2 $k = 10$	**11.718**	0.0	**11.718**	0.0	11.605	4.1
CovColon $k = 10$	**2641.228**	0.4	**2641.228**	0.4	**2641.228**	59.7
Lymp $k = 10$	**5911.412**	0.3	**5911.412**	0.2	5753.563	81.4
Reddit $k = 10$	**1052.020**	7.4	**1052.020**	4.5	**1052.020**	1881.4

solutions with more or less the same objective function values.

A.9 Comparison with Existing Methods for Dual Bounds

In this section, we compare the performance of our convex integer program method with (1) Mosek, in our experience one of the best commercial implementations of SDP solvers; and (2) two variants of the approach presented in [39], which uses the main idea of [119].

The variants are listed as follows:

A.9.1 Dual Alternating Direction Augmented Lagrangian (DADAL) Method

Dual Alternating Direction Augmented Lagrangian (DADAL) method [39] can be used to find out the upper bounds of the SDP problem. In order to use the freely available implementation, the DADAL method requires the remodeling of the original problem into the following standard format:

$$\min\langle \boldsymbol{A}, \boldsymbol{V}\rangle \text{ s.t } \mathcal{A}(\boldsymbol{V}) = \boldsymbol{b},\ \boldsymbol{V} \succeq \boldsymbol{0}.$$

Thus to find the dual bounds of the sparse PCA with covariance matrix of size d, we need to (1) add additional auxiliary variables for inequality constraints, (2) reformulate the variables into a p.s.d. matrix. For the step-(1), the original sparse PCA problem can be formulated in the following fashion:

$$\begin{aligned}
&\min \langle -\boldsymbol{A}, \boldsymbol{V}\rangle && \text{(SDP-equality)}\\
&\text{s.t. } \langle \boldsymbol{I}_d, \boldsymbol{V}\rangle + \mu_1 = 1\\
&\quad \langle \boldsymbol{I}_{d^2}, \text{diag}(\boldsymbol{Y})\rangle + \mu_2 = k\\
&\quad \langle \boldsymbol{E}_{ij}^{+}, \boldsymbol{V} \oplus \text{diag}(\boldsymbol{Y})\rangle + \gamma_{ij}^{+} = 0,\ \forall\, ij\\
&\quad \langle \boldsymbol{E}_{ij}^{-}, \boldsymbol{V} \oplus \text{diag}(\boldsymbol{Y})\rangle + \gamma_{ij}^{-} = 0,\ \forall\, ij\\
&\quad \boldsymbol{V}, \text{diag}(\boldsymbol{Y}), \text{diag}(\boldsymbol{\gamma}^{+}), \text{diag}(\boldsymbol{\gamma}^{-}), \text{diag}(\mu) \succeq \boldsymbol{0}
\end{aligned}$$

where $\oplus$ is the direct sum of two matrices, i.e., $\boldsymbol{A} \oplus \boldsymbol{B} := \begin{pmatrix} \boldsymbol{A} & 0 \\ 0 & \boldsymbol{B} \end{pmatrix}$, the matrix $\text{diag}(\boldsymbol{Y})$ is a short notation of $\text{diag}(\text{vec}(\boldsymbol{Y}))$ with $\text{vec}(\boldsymbol{Y})$ the vectorization of matrix $\boldsymbol{Y}$, and the

matrix $\boldsymbol{E}_{ij}^{+}, \boldsymbol{E}_{ij}^{-}$ are

$$\boldsymbol{E}_{ij}^{+} := \begin{pmatrix} E_{ij} & 0 \\ 0 & -\text{diag}(\text{vec}(E_{ij})) \end{pmatrix}, \ \boldsymbol{E}_{ij}^{-} := \begin{pmatrix} -E_{ij} & 0 \\ 0 & -\text{diag}(\text{vec}(E_{ij})) \end{pmatrix}, \quad \forall i,j \in [d] \times [d]$$

with $E_{ij} \in \mathbb{R}^{d \times d}$ the standard basis matrix (i.e., the component (i,j) equals to 1, and the rest components equal to 0). Rewrite the variables of SDP-equality into a p.s.d. matrix

$$\tilde{\boldsymbol{V}} := \begin{pmatrix} \boldsymbol{V} & & & & \\ & \text{diag}(\boldsymbol{Y}) & & & \\ & & \text{diag}(\boldsymbol{\gamma}^{+}) & & \\ & & & \text{diag}(\boldsymbol{\gamma}^{-}) & \\ & & & & \mu \end{pmatrix} \in \mathbb{R}^{(d+3d^2+2)\times(d+3d^2+2)}.$$

For the step-(2), the SDP-equality can be further transferred into the standard SDP format as follows:

$$\begin{aligned} \min \ & \langle -\boldsymbol{A} \oplus \boldsymbol{0}_{d^2} \oplus \boldsymbol{0}_{d^2} \oplus \boldsymbol{0}_{d^2} \oplus \boldsymbol{0}_2, \tilde{\boldsymbol{V}} \rangle && \text{(standard-SDP)} \\ \text{s.t.} \ & \langle \boldsymbol{I}_d \oplus \boldsymbol{0}_{d^2} \oplus \boldsymbol{0}_{d^2} \oplus \boldsymbol{0}_{d^2} \oplus \text{diag}(1,0), \tilde{\boldsymbol{V}} \rangle = 1 \\ & \langle \boldsymbol{0}_d \oplus \boldsymbol{I}_{d^2} \oplus \boldsymbol{0}_{d^2} \oplus \boldsymbol{0}_{d^2} \oplus \text{diag}(0,1), \tilde{\boldsymbol{V}} \rangle = k \\ & \langle (\boldsymbol{E}_{ij}^{+} + \boldsymbol{E}_{ij}^{+}) \oplus (\text{diag}(\text{vec}(E_{ij})) + \text{diag}(\text{vec}(E_{ji}))) \oplus \boldsymbol{0}_{d^2} \oplus \boldsymbol{0}_2, \tilde{\boldsymbol{V}} \rangle = 0, \ \forall i \geq j \\ & \langle (\boldsymbol{E}_{ij}^{-} + \boldsymbol{E}_{ij}^{-}) \oplus \boldsymbol{0}_{d^2} \oplus (\text{diag}(\text{vec}(E_{ij})) + \text{diag}(\text{vec}(E_{ji}))) \oplus \boldsymbol{0}_2, \tilde{\boldsymbol{V}} \rangle = 0, \ \forall i \geq j \\ & \tilde{\boldsymbol{V}} \succeq \boldsymbol{0} \end{aligned}$$

with the size of variable matrix $d+3d^2+2$ and the number of linear constraints $2+d\times(d+1)$. The code of DADAL method is downloaded from the author's [39] homepage [1].

[1] https://www.math.aau.at/or/Software/

A.9.2 A Variant of DADAL Method for SPCA (DADAL-SPCA)

In this subsection, we designed a DADAL-SPCA method (which uses the main ideas of the DADAL method) that works specifically for the sparse PCA problem. As we have seen above, using the standard code of DADAL involves increasing dimension to $(d+3d^2+2)^2$ which appears to be quiet inefficient for solving the standard SDP relaxation of sparse PCA. Therefore we alternatively pursued the following approach: Consider the primal and dual SDP relaxation of sparse PCA,

$$
\begin{array}{lll}
\min_{\boldsymbol{V},\boldsymbol{Y}} & \langle -\boldsymbol{A}, \boldsymbol{V}\rangle =: \text{Primal} & \\
\text{s.t.} & \langle \boldsymbol{I}, \boldsymbol{V}\rangle \le 1 & (\mu_1 \ge 0) \\
& \langle \mathbf{1}\mathbf{1}^\top, \boldsymbol{Y}\rangle \le k & (\mu_2 \ge 0) \\
& \boldsymbol{Y} \succeq \boldsymbol{V} & (\boldsymbol{\gamma}^+ \ge 0) \\
& \boldsymbol{Y} \succeq -\boldsymbol{V} & (\boldsymbol{\gamma}^- \ge 0) \\
& \boldsymbol{V} \succeq \mathbf{0} & (\boldsymbol{Z} \succeq \mathbf{0})
\end{array}
\qquad
\begin{array}{ll}
\max & -\mu_1 - \mu_2 k =: \text{Dual} \\
\text{s.t.} & \mu_1 \boldsymbol{I} + \boldsymbol{\gamma}^+ - \boldsymbol{\gamma}^- - \boldsymbol{A} - \boldsymbol{Z} = \mathbf{0} \\
& \mu_2 \mathbf{1}\mathbf{1}^\top - \boldsymbol{\gamma}^+ - \boldsymbol{\gamma}^- = \mathbf{0} \\
& \boldsymbol{Z} \succeq \mathbf{0} \\
& \mu_1, \mu_2, \boldsymbol{\gamma}^+, \boldsymbol{\gamma}^- \ge 0
\end{array}
$$

with its augmented Lagrangian

$$
\mathcal{L}_\sigma(\boldsymbol{\mu}, \boldsymbol{\gamma}, \boldsymbol{Z}; \boldsymbol{V}, \boldsymbol{Y}) := -\mu_1 - \mu_2 k + \langle \boldsymbol{M}_1, \boldsymbol{X}\rangle + \langle \boldsymbol{M}_2, \boldsymbol{Y}\rangle - \frac{\sigma}{2}\|\boldsymbol{M}_1\|_F^2 - \frac{\sigma}{2}\|\boldsymbol{M}_2\|_F^2,
$$

where $\boldsymbol{M}_1, \boldsymbol{M}_2$ are defined as

$$
\begin{aligned}
\boldsymbol{M}_1 &:= \mu_1 \boldsymbol{I} + \boldsymbol{\gamma}^+ - \boldsymbol{\gamma}^- - \boldsymbol{A} - \boldsymbol{Z}, \\
\boldsymbol{M}_2 &:= \mu_2 \mathbf{1}\mathbf{1}^\top - \boldsymbol{\gamma}^+ - \boldsymbol{\gamma}^-.
\end{aligned}
$$

We initialize $\boldsymbol{V}^0, \boldsymbol{Y}^0, \boldsymbol{Z}^0$ as follows: Compute eigenvalue decomposition of $\boldsymbol{A} = \boldsymbol{W}\boldsymbol{\Lambda}_{\boldsymbol{A}}\boldsymbol{W}^\top$, let $\boldsymbol{w}_1$ be the leading eigenvector of $\boldsymbol{W}$ with respect to the largest eigenvalue. Set

$$\begin{aligned}\boldsymbol{V}^0 &\leftarrow \boldsymbol{w}_1\boldsymbol{w}_1^\top,\\ \boldsymbol{Y}^0 &\leftarrow |\boldsymbol{V}^0|,\\ \boldsymbol{Z}^0 &\leftarrow \boldsymbol{0},\end{aligned}$$

along with the starting augmented Lagrangian parameter σ^0. In $(k+1)$-th iteration, update each variable based on the following rule which is similar as the DADAL method proposed in [39].

$$\begin{aligned}\boldsymbol{\mu}^{k+1}, \boldsymbol{\gamma}^{k+1} &\leftarrow \underset{\boldsymbol{\mu}\geq 0, \boldsymbol{\gamma}\geq 0}{\arg\max}\, \mathcal{L}_{\sigma^k}(\boldsymbol{\mu}, \boldsymbol{\gamma}, \boldsymbol{Z}^k; \boldsymbol{V}^k, \boldsymbol{Y}^k)\\ \boldsymbol{Z}^{k+1} &\leftarrow \left(-\frac{\boldsymbol{V}^k}{\sigma^k} + \boldsymbol{\mu}_1^{k+1}\boldsymbol{I} + (\boldsymbol{\gamma}^+)^{k+1} - (\boldsymbol{\gamma}^-)^{k+1} - \boldsymbol{A}\right)_{\succeq 0}\\ \boldsymbol{X}^{k+1} &\leftarrow -\sigma \cdot \left(-\frac{\boldsymbol{V}^k}{\sigma^k} + \boldsymbol{\mu}_1^{k+1}\boldsymbol{I} + (\boldsymbol{\gamma}^+)^{k+1} - (\boldsymbol{\gamma}^-)^{k+1} - \boldsymbol{A}\right)_{\preceq 0}\\ \boldsymbol{Y}^{k+1} &\leftarrow |\boldsymbol{V}^{k+1}|\end{aligned}$$

Update σ based on Algorithm 1 in [39]

where $(\boldsymbol{A})_{\succeq 0}, (\boldsymbol{A})_{\preceq 0}$ denote the positive semi-definite, negative semi-definite part of symmetric matrix $\boldsymbol{A}$. That is: Let $\boldsymbol{A} = \boldsymbol{W}\boldsymbol{\Lambda}_{\boldsymbol{A}}\boldsymbol{W}^\top$ be its eigenvalue decomposition. Represent $\boldsymbol{\Lambda}_{\boldsymbol{A}} = \boldsymbol{\Lambda}_{\boldsymbol{A}}^+ + \boldsymbol{\Lambda}_{\boldsymbol{A}}^-$ where $[\boldsymbol{\Lambda}_{\boldsymbol{A}}^+]_{ii} = \max\{[\boldsymbol{\Lambda}_{\boldsymbol{A}}]_{ii}, 0\}$ and $[\boldsymbol{\Lambda}_{\boldsymbol{A}}^-]_{ii} = \min\{[\boldsymbol{\Lambda}_{\boldsymbol{A}}]_{ii}, 0\}$, then

$$\begin{aligned}(\boldsymbol{A})_{\succeq 0} &:= \boldsymbol{W}\boldsymbol{\Lambda}_{\boldsymbol{A}}^+\boldsymbol{W}^\top,\\ (\boldsymbol{A})_{\preceq 0} &:= \boldsymbol{W}\boldsymbol{\Lambda}_{\boldsymbol{A}}^-\boldsymbol{W}^\top.\end{aligned}$$

Remark A.9.1. *The way we update our dual variables (and primal variables) in each iteration, there is no guarantee that the dual variables satisfy the equality constraints in*

the dual, namely,

$$\boldsymbol{M}_1 := \mu_1 \boldsymbol{I} + \boldsymbol{\gamma}^+ - \boldsymbol{\gamma}^- - \boldsymbol{A} - \boldsymbol{Z} = 0,$$
$$\boldsymbol{M}_2 := \mu_2 \mathbf{1}\mathbf{1}^\top - \boldsymbol{\gamma}^+ - \boldsymbol{\gamma}^- = 0.$$

Therefore, it is not true that we can always obtain exact dual bounds from every iteration. We store the dual bounds of iterations where the equality constraints are satisfied within a tolerance of 0.01*, i.e.,*

$$\|\boldsymbol{M}_1\|_F + \|\boldsymbol{M}_2\|_F \leq 0.01.$$

Moreover, after the final iteration, we add one more step by solving the following linear program*,*

$$\begin{aligned} \mu^{\text{final}}, \boldsymbol{\gamma}^{\text{final}} := \arg\max_{\mu,\boldsymbol{\gamma}} \quad & -\mu_1 - \mu_2 k \\ \textit{s.t.} \quad & \mu_1 \boldsymbol{I} + \boldsymbol{\gamma}^+ - \boldsymbol{\gamma}^- - \boldsymbol{A} - \boldsymbol{Z}^{\text{final}} = \mathbf{0}, \\ & \mu_2 \mathbf{1}\mathbf{1}^\top - \boldsymbol{\gamma}^+ - \boldsymbol{\gamma}^- = \mathbf{0}, \\ & \mu_1, \mu_2, \boldsymbol{\gamma}^+, \boldsymbol{\gamma}^- \geq 0, \end{aligned} \tag{final-dual}$$

where $\boldsymbol{Z}^{\text{final}} \succeq 0$ *is the dual variable obtained in the final step of DADAL-SPCA. It is easy to observe that* $(\mu^{\text{final}}, \boldsymbol{\gamma}^{\text{final}}, \boldsymbol{Z}^{\text{final}})$ *is a dual feasible solution, and therefore a dual bound can be obtained from this dual feasible solution.*

Stopping criteria: The stopping criteria includes three conditions. Meeting any of the criteria stops the DADAL-SPCA algorithm.

1. The maximum number of iteration is set to be 200.

2. The stopping criteria quantity δ proposed in Algorithm 1 [39] is set to be 0.001, i.e., at the end of each iteration, we compute the primal and dual infeasibility errors as

follows:

$$r_P := \frac{\max\{\text{Tr}(X) - 1, 0\} + \max\{\langle \mathbf{1}\mathbf{1}^\top, \boldsymbol{Y}\rangle - k, 0\}}{1 + \sqrt{1 + k^2}},$$
$$r_D := \frac{\|\boldsymbol{M}_1\|_F + \|\boldsymbol{M}_2\|_F}{1 + \|\boldsymbol{A}\|_F},$$

and set $\delta := \max\{r_P, r_D\}$.

3. Since there is no closed form solution of the following updating step:

$$\boldsymbol{\mu}^{k+1}, \boldsymbol{\gamma}^{k+1} \leftarrow \underset{\boldsymbol{\mu}\geq 0, \boldsymbol{\gamma}\geq 0}{\arg\max} \mathcal{L}_{\sigma^k}(\boldsymbol{\mu}, \boldsymbol{\gamma}, \boldsymbol{Z}^k; \boldsymbol{V}^k, \boldsymbol{Y}^k),$$

we use commercial solver Gurobi (called via Python) to solve this quadratic programming sub-problem in each iteration. For small instances (i.e., $d < 500$, Pitprops, Eisen-1, Eisen-2), the total time limit given for Gurobi solver is 3600 seconds (1 hour); and for middle-size instance (i.e., $d = 500$, CovColon, Lymp), the total time limit given for Gurobi solver is 7200 seconds (2 hours), and for large instance (i.e., $d = 2000$, Reddit), the total time limit given for Gurobi solver is 18000 seconds (5 hours).

Algorithm 7 is the pseudocode of finding dual bounds using DADAL-SPCA.

Algorithm 7 Dual Bound DADAL-SPCA

Input: Covariance matrix $\boldsymbol{A}$, sparsity parameter k, maximum number of iteration $T_{\max}$, total time limit for solver T_{total}, starting Lagrangian augmented parameter σ^0.
Output: Dual bound of sparse PCA.
function DUAL BOUND METHOD($\boldsymbol{A}, k, T_{\max}, T_{\text{total}}$)
    Compute eigenvalue decomposition on $\boldsymbol{A}$, let $\boldsymbol{w}_1$ be its leading eigenvector.
    Initialize $\boldsymbol{V} \leftarrow \boldsymbol{w}_1\boldsymbol{w}_1^\top, \boldsymbol{Y} \leftarrow |\boldsymbol{V}|, \boldsymbol{Z} \leftarrow \mathbf{0}^{d\times d}, (\mu_1, \mu_2) \leftarrow (0, 0), \boldsymbol{\gamma}^{\pm} \leftarrow \mathbf{0}^{d\times d}$.
    Run DADAL-SPCA with stopping criteria described above with starting Lagrangian augmented parameter $\sigma^0 \in \{0.001, 0.01, 0.1, 1\}$, and return $\text{UB}^{\text{DADAL-SPCA}}$.
    Solve final-dual for a dual bound $\text{UB}^{\text{final-dual}}$.
    return $\text{UB} \leftarrow \min\{\text{UB}^{\text{final-dual}}, \text{UB}^{\text{DADAL-SPCA}}\}$.
end function

A.9.3 Numerical Results for Existing Methods for Dual Bounds

The gap obtained by DADAL-SPCA as described above with various values of σ^0 is reported in Table A.2.

Table A.2: DADAL-SPCA under different starting augmented Lagrangian parameter σ^0.

Instance \ σ^0	LB	$\sigma^0 = 0.001$		$\sigma^0 = 0.01$		$\sigma^0 = 0.1$		$\sigma^0 = 1$	
		gap %	Time	gap %	Time	gap %	Time	gap %	Time
Pitprops $k = 5$	3.406	3.96	6	1.79	5	1.70	2	**1.64**	3
Eisen-1 $k = 10$	17.33	2.23	270	**2.19**	225	11.07	294	39.10	288
Eisen-2 $k = 10$	11.71	2.32	1053	2.37	610	**2.08**	898	2.12	897
CovColon $k = 10$	2641	14.16	7492	**13.51**	7281	19.05	7369	26.82	7301
Lymp $k = 10$	6008	**29.67**	7339	34.79	7331	46.84	7367	59.09	7373
Reddit $k = 10$	1052	-	O.M.	-	O.M.	-	O.M.	-	O.M.

The "Time" column in Table A.2 denotes the total running time used for the DADAL-SPCA method. We can see that the "Time" of CovColon, Lymp reported in Table A.2 are greater than time limit for solver, since additional time are required to implement the other four updating steps in each iteration. The out of memory (O.M.) for Reddit instance is due to the memory limitation to load Reddit instance $d = 2000$ for the update step

$$\boldsymbol{\mu}^{k+1}, \boldsymbol{\gamma}^{k+1} \leftarrow \underset{\boldsymbol{\mu}\geq 0, \boldsymbol{\gamma}\geq 0}{\arg\max} \mathcal{L}_{\sigma^k}(\boldsymbol{\mu}, \boldsymbol{\gamma}, \boldsymbol{Z}^k; \boldsymbol{V}^k, \boldsymbol{Y}^k).$$

We tried to solve the final-dual linear program for Reddit instance, but the LP did not solve in 5 hours. (This LP has order d^2 variables, whereas the number of variables of convex integer program is order $dI_{\text{pos}}N$ and $I_{\text{pos}}N \ll d$ in this instance.)

To complete the comparison, we also list the comparison between our model in paper and DADAL, DADAL-SPCA, Mosek in Table A.3.

Based on Table A.3, we observe that the SDP-relaxation solved by Mosek produces the best bounds for the small instances (Pitprops, Eisen-1, Eisen-2), while DADAL-SPCA is able to produce bounds for Pitprops, Eisen-1, Eisen-2, CovColon, and Lymp. However, as we can see, except for Pitprops, the best dual bounds are obtained by solving convex IP model of this paper.

Table A.3: Compare with existing SDP methods

Instance	LB	Model-in-Paper		DADAL [39]		DADAL-SPCA (best)		Mosek	
		gap %	Time	gap %	Time	gap %	Time	gap %	Time
Pitprops $k = 5$	3.406	3.26	0.4	82.43	593	1.64	3	**1.52**	5
Eisen-1 $k = 10$	17.33	**0.115**	63	-	O.M.	2.19	225	2.19	15
Eisen-2 $k = 10$	11.71	**1.71**	385	-	O.M.	2.08	898	1.96	52
CovColon $k = 10$	2641	**2.37**	28	-	O.M.	13.51	7281	-	O.M.
Lymp $k = 10$	6008	**17.86**	4225	-	O.M.	29.67	7339	-	O.M.
Reddit $k = 10$	1052	**2.24**	8584	-	O.M.	-	O.M.	-	O.M.

APPENDIX B

APPENDICES FOR CHAPTER 2

B.1 Additional concentration inequalities

We need the standard multiplicative Chernoff bound (see Theorem 4.4 [74]).

Lemma B.1.1 (Chernoff Bound)**.** *Let $X_1, \ldots, X_n$ be independent random variables taking values in $[0, 1]$. Then for any $\delta > 0$ we have*

$$\Pr\left(\sum_i X_i > (1+\delta)\mu\right) < \left(\frac{e}{1+\delta}\right)^{(1+\delta)\mu},$$

where $\mu = \mathbb{E}\sum_i X_i$.

We also need the one-sided Chebychev inequality, see for example Exercise 3.18 of [74].

Lemma B.1.2 (One-sided Chebychev)**.** *For any random variable X with finite first and second moments*

$$\Pr\left(X \leq \mathbb{E}X - t\right) \leq \frac{\mathit{Var}(X)}{\mathit{Var}(X) + t^2}.$$

B.2 Techniques for reducing the running time of CIP

In practice, we want to reduce the running time of CIP. Here are the techniques that we used to enhance the efficiency in practice.

B.2.1 Threshold

The first technique is to reduce the number of SOS-II constraints. Let λ_{TH} be a threshold parameter that splits the eigenvalues $\{\lambda_j\}_{j=1}^d$ of sample covariance matrix $\boldsymbol{A}$ into two parts

$J^+ = \{j : \lambda_j > \lambda_{\text{TH}}\}$ and $J^- = \{j : \lambda_j \leq \lambda_{\text{TH}}\}$. The objective function $\text{Tr}\left(\boldsymbol{V}^\top \boldsymbol{A}\boldsymbol{V}\right)$ satisfies

$$\text{Tr}\left(\boldsymbol{V}^\top \boldsymbol{A}\boldsymbol{V}\right) = \sum_{j\in J^+}(\lambda_j - \lambda_{\text{TH}})\sum_{i=1}^{r} g_{ji}^2 + \sum_{j\in J^-}(\lambda_j - \lambda_{\text{TH}})\sum_{i=1}^{r} g_{ji}^2 + \lambda_{\text{TH}}\sum_{j=1}^{d}\sum_{i=1}^{r} g_{ji}^2,$$

in which the first term is convex, the second term is concave, and the third term satisfies

$$\lambda_{\text{TH}}\sum_{j=1}^{d}\sum_{i=1}^{r} g_{ji}^2 \leq r\lambda_{\text{TH}} \qquad \text{(threshold-term)}$$

due to $\sum_{j=1}^{d}\sum_{i=1}^{r} g_{ji}^2 \leq r$. Since maximizing a concave function is equivalent to convex optimization, we replace the second term by a new auxiliary variable s and the third term by its upper bound $r\lambda_{\text{TH}}$ such that

$$\text{Tr}\left(\boldsymbol{V}^\top \boldsymbol{A}\boldsymbol{V}\right) \leq \sum_{j\in J^+}(\lambda_j - \lambda_{\text{TH}})\sum_{i=1}^{r} g_{ji}^2 - s + r\lambda_{\text{TH}} \qquad \text{(threshold-tech)}$$

where

$$s \geq \sum_{j\in J^-} \underbrace{(\lambda_{\text{TH}} - \lambda_j)}_{\geq 0}\sum_{i=1}^{r} g_{ji}^2 \qquad \text{(s-var)}$$

is a convex constraint.

We select a value of λ_{TH} so that $|J^+| = 3$. Therefore, it is sufficient to construct a piecewise-linear upper approximation for the quadratic terms g_{ji}^2 in the first term with $j \in J^+$, i.e., constraint set PLA($[J^+] \times [r]$). We thus, greatly reduce the number of SOS-II constraints from $\mathcal{O}(d \times r)$ to $\mathcal{O}(|J^+| \times r)$, i.e. in our experiemnts to $3r$ SOS-II constraints.

B.2.2 Cutting planes

Similar to classical integer programming, we can incorporate additional cutting planes to improve the efficiency.

Cutting plane for sparsity: The first family of cutting-planes is obtained as follows: Since $\|\boldsymbol{V}\|_0 \leq k$ and $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r$ are orthogonal, by Bessel inequality, we have

$$\sum_{i=1}^{r} g_{ji}^2 = \sum_{i=1}^{r} (\boldsymbol{w}_j^\top \boldsymbol{v}_i)^2 = \boldsymbol{w}_j^\top \boldsymbol{V}\boldsymbol{V}^\top \boldsymbol{w}_j \leq \theta_j^2, \tag{sparse-g}$$

$$\sum_{i=1}^{r} \xi_{ji} \leq \theta_j^2 \left(1 + \frac{r}{4N^2}\right). \tag{sparse-xi}$$

We call these above cuts–sparse cut since θ_j is obtained from the row sparsity parameter k.

Cutting plane from objective value: The second type of cutting plane is based on the property: for any symmetric matrix, the sum of its diagonal entries are equal to the sum of its eigenvalues. Let $\boldsymbol{A}_{j_1,j_1}, \ldots, \boldsymbol{A}_{j_k,j_k}$ be the largest k diagonal entries of the sample covariance matrix $\boldsymbol{A}$, we have

Proposition B.2.1. *The following are valid cuts for rsPCA:*

$$\sum_{j=1}^{d} \lambda_j \sum_{i=1}^{r} g_{ji}^2 \leq \boldsymbol{A}_{j_1,j_1} + \cdots + \boldsymbol{A}_{j_k,j_k}. \tag{cut-g}$$

When the splitting points $\{\gamma_{ji}^\ell\}_{\ell=-N}^{N}$ in SOS-II are set to be $\gamma_{ji}^\ell = \frac{\ell}{N} \cdot \theta_j$, we have:

$$\begin{aligned} \textstyle\sum_{j\in J^+} (\lambda_j - \lambda_{\mathrm{TH}}) \sum_{i=1}^{r} \xi_{ji} - s + g\lambda_{\mathrm{TH}} &\leq \boldsymbol{A}_{j_1,j_1} + \cdots + \boldsymbol{A}_{j_k,j_k} + \textstyle\sum_{j\in J^+} \frac{r(\lambda_j - \phi)\theta_j^2}{4N^2} \\ g &\geq \textstyle\sum_{j=1}^{d} \sum_{i=1}^{r} g_{ji}^2. \end{aligned} \tag{cut-xi}$$

Implemented version of CIP

Thus the implemented version of CIP is

$$\begin{array}{ll} \max & \sum_{j\in J^+}(\lambda_j - \lambda_{\mathrm{TH}})\sum_{i=1}^{r}\xi_{ji} - s + r\lambda_{\mathrm{TH}} \\ \text{s.t} & \boldsymbol{V} \in \mathcal{CR}2 \\ & (g,\xi,\eta) \in \mathrm{PLA}' \\ & \text{(s-var), (sparse-g), (sparse-xi), (cut-g), (cut-xi)} \end{array} \tag{CIP-impl}$$

B.2.3 Submatrix technique

Proposition B.2.2. *Given any matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$, let θ be defined as*

$$\theta := 2\mathrm{max}_{\boldsymbol{V}^1\in\mathbb{R}^{m\times r},\boldsymbol{V}^2\in\mathbb{R}^{n\times r}}\ 2\mathrm{Tr}\left((\boldsymbol{V}^1)^\top \boldsymbol{X}\boldsymbol{V}^2\right)\ \textit{s.t.}\ (\boldsymbol{V}^1)^\top\boldsymbol{V}^1 + (\boldsymbol{V}^2)^\top\boldsymbol{V}^2 = \boldsymbol{I}^r,$$

then $\theta \le \sqrt{r}\|\boldsymbol{X}\|_F$

Proof.

$$\begin{aligned} & \max_{\boldsymbol{V}^1,\boldsymbol{V}^2}\ 2\mathrm{Tr}\left((\boldsymbol{V}^1)^\top \boldsymbol{X}\boldsymbol{V}^2\right)\ \text{s.t.}\ (\boldsymbol{V}^1)^\top\boldsymbol{V}^1 + (\boldsymbol{V}^2)^\top\boldsymbol{V}^2 = \boldsymbol{I}^r, \\ \Leftrightarrow\quad & \max_{\boldsymbol{V}^1,\boldsymbol{V}^2}\ \mathrm{Tr}\left(\begin{pmatrix}(\boldsymbol{V}^1)^\top & (\boldsymbol{V}^2)^\top\end{pmatrix}\begin{pmatrix}0 & \boldsymbol{X}\\ \boldsymbol{X}^\top & 0\end{pmatrix}\begin{pmatrix}\boldsymbol{V}^1\\ \boldsymbol{V}^2\end{pmatrix}\right)\ \text{s.t.}\ (\boldsymbol{V}^1)^\top\boldsymbol{V}^1 + (\boldsymbol{V}^2)^\top\boldsymbol{V}^2 = \boldsymbol{I}^r, \\ \Leftrightarrow\quad & \max_{\boldsymbol{V}}\ \mathrm{Tr}\left(\boldsymbol{V}^\top\begin{pmatrix}0 & \boldsymbol{X}\\ \boldsymbol{X}^\top & 0\end{pmatrix}\boldsymbol{V}\right)\ \text{s.t.}\ \boldsymbol{V}^\top\boldsymbol{V} = \boldsymbol{I}^r. \end{aligned}$$

Note that the final maximization problem is equal to

$$\max_{\boldsymbol{V}} \text{ Tr} \left(\boldsymbol{V}^\top \begin{pmatrix} 0 & \boldsymbol{X} \\ \boldsymbol{X}^\top & 0 \end{pmatrix} \boldsymbol{V} \right) \text{ s.t. } \boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{I}^r$$
$$\leq \sum_{i=1}^{r} \lambda_i \left(\begin{pmatrix} 0 & \boldsymbol{X} \\ \boldsymbol{X}^\top & 0 \end{pmatrix} \right),$$

Next we verify that the eigenvalues of

$$\begin{pmatrix} 0 & \boldsymbol{X} \\ \boldsymbol{X}^\top & 0 \end{pmatrix}$$

are $\pm$ singular values of $\boldsymbol{X}$: Let $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^\top$. In particular, note that:

$$\begin{pmatrix} 0 & \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^\top \\ \boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{U}^\top & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_i \\ \boldsymbol{w}_i \end{pmatrix} = \begin{pmatrix} \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{e}_i \\ \boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{e}_i \end{pmatrix} = \sigma_i(\boldsymbol{X}) \begin{pmatrix} \boldsymbol{u}_i \\ \boldsymbol{w}_i \end{pmatrix}$$
$$\begin{pmatrix} 0 & \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^\top \\ \boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{U}^\top & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_i \\ -\boldsymbol{w}_i \end{pmatrix} = \begin{pmatrix} -\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{e}_i \\ \boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{e}_i \end{pmatrix} = -\sigma_i(\boldsymbol{X}) \begin{pmatrix} \boldsymbol{u}_i \\ -\boldsymbol{w}_i \end{pmatrix}.$$

Therefore, we have

$$\sum_{i=1}^{r} \lambda_i \left(\begin{pmatrix} 0 & \boldsymbol{X} \\ \boldsymbol{X}^\top & 0 \end{pmatrix} \right) = \sum_{i=1}^{r} \sigma_i(\boldsymbol{X}) \leq \sqrt{r} \|\boldsymbol{X}\|_F.$$

□

APPENDIX C

APPENDICES FOR CHAPTER 3

C.1 Proof of Proposition 3.2.1

Proof. Since

$$\phi(\boldsymbol{\beta}, \beta_0) = \sum_{i \in \{m+1,\ldots,n\}} (\max\{0, \boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0\} - Y_i)^2,$$

then it is sufficient to show that $(\max\{0, \boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0\} - Y_i)^2$ is convex for each $i = m + 1, \ldots, n$. Let $\theta(x) = (\max\{0, x\} - Y_i)^2 = (\max\{0, x\})^2 + Y_i^2 - 2Y_i \max\{0, x\}$ with $Y_i < 0$. Note that $\theta(x)$ is convex over $x \in \mathbb{R}$. Let $L(\boldsymbol{\beta}, \beta_0) = \boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0$ be an affine function. Then $(\max\{0, \boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0\} - Y_i)^2 = \theta(L(\boldsymbol{\beta}, \beta_0))$ is convex. □

C.2 Proof of Theorem 8

In order to prove Theorem 8, we show that the subset sum problem can be polynomially reduced to a special case of ReLU-regression problem. We begin a definition of the subset sum problem.

Definition C.2.1. ***Subset sum problem:*** *Given p non-negative integers $a_1, \ldots, a_d$, the subset sum problem is to find out whether there exists a subset $S \subseteq [d]$ such that $\sum_{i \in S} a_i = \frac{1}{2} \sum_{i=1}^d a_i$.*

Note that the subset sum problem is equivalent to find out a feasible solution $x \in \{0, 1\}^d$ such that $\sum_{i=1}^d a_i x_i = \frac{1}{2} \sum_{i=1}^d a_i$. Therefore, the following $\{\pm 1\}-$subset sum problem is still NP-hard.

Definition C.2.2. $\{\pm 1\}-$***subset sum problem:*** *Given d nonnegative integers $a_1, \ldots, a_d$, the $\{\pm 1\}-$subset sum problem is to decide if there exists a solution $x \in \{\pm 1\}^d$ such that*

$\sum_{i=1}^{d} a_i x_i = \frac{1}{2}\sum_{i=1}^{d} a_i$.

Proposition C.2.1. *The decision problem $\{\pm 1\}-$subset sum problem is in NP-complete.*

Proof. Clearly, $\{\pm 1\}-$subset sum problem is in NP. In order to show that $\{\pm 1\}-$**subset sum problem** is in NP-complete, we show that the instance of subset sum corresponding to $(a_1, \ldots, a_d)$ is feasible if and only if the $\{\pm 1\}-$**subset sum** instance $(a_1, \ldots, a_d, a_{d+1})$ with $a_{d+1} = \sum_{i=1}^{d} a_i$ is feasible.

Clearly if the subset set instance is feasible, then there exists a subset $S \subseteq [d]$ such that $\sum_{i\in S} a_i = \frac{1}{2}\sum_{i=1}^{d} a_i$. Then setting $x_i = 1$ for $i \in S \cup \{d+1\}$ and $x_i = -1$ for $i \in [d] \setminus S$ gives us: $\sum_{i=1}^{d+1} a_i x_i = \frac{1}{2}\sum_{i=1}^{d+1} a_i$.

On the other hand if the $\{\pm 1\}-$subset sum is feasible, there exists some $x_i \in \{-1, 1\}^{d+1}$ such that $\sum_{i=1}^{d+1} a_i x_i = \frac{1}{2}\sum_{i=1}^{d+1} a_i$. First observe that x_{d+1} cannot be -1 since then we would have that $\sum_{i=1}^{d} a_i x_i = 2\sum_{i=1}^{d} a_i$. Thus, we have that $\sum_{i=1}^{d} a_i x_i = 0$ implying that there exists $S \subseteq [d]$ such that $\sum_{i\in S} a_i = \frac{1}{2}\sum_{i=1}^{d} a_i$. □

Now we show the equivalence between $\{\pm 1\}-$subset sum problem and a special case of ReLU-regression problem. Consider the following auxiliary function $\theta(x, \beta_0)$ defined as:

$$(\max\{0, x + \beta_0\} - 1)^2 + (\max\{0, -x + \beta_0\} - 1)^2$$

(See Figure C.1). For a fixed x, let $g(\beta_0) = \min_x \theta(x, \beta_0)$. Let $\tau(\boldsymbol{\beta}, \beta_0)$ be

$$\tau(\boldsymbol{\beta}, \beta_0) := \left(\max\left\{0, \sum_{i=1}^{p} a_i \boldsymbol{\beta}_i + \beta_0\right\} - \frac{1}{2}\sum_{i=1}^{p} a_i\right)^2 + \left(\max\left\{0, \sum_{i=1}^{p} 2 \cdot a_i \boldsymbol{\beta}_i + \beta_0\right\} - \sum_{i=1}^{p} a_i\right)^2.$$

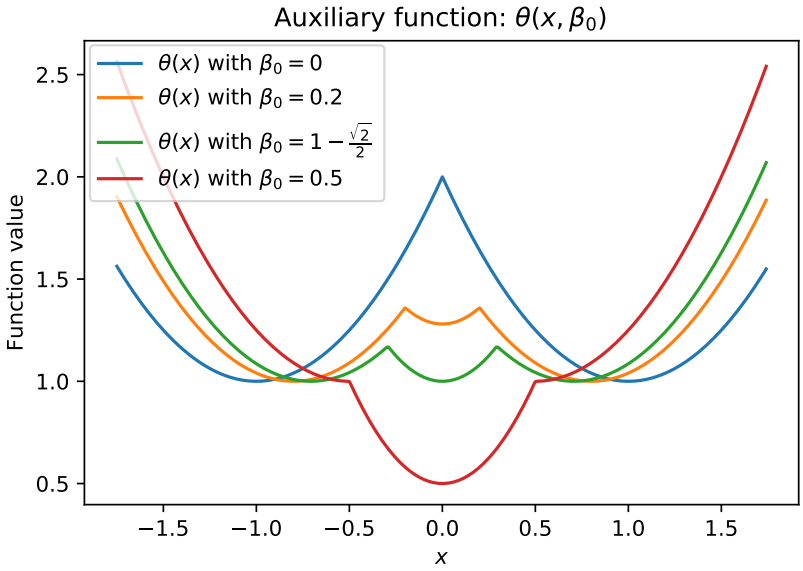


Figure C.1: Function $\theta(x, \beta_0)$

We construct our affine ReLU-regression problem as follows:

$$\min_{\boldsymbol{\beta}, \beta_0 \in \mathbb{R}^{d+1}} \tau(\boldsymbol{\beta}, \beta_0) + \sum_{i=1}^{d} \theta(\boldsymbol{e}_j^\top \boldsymbol{\beta}, \beta_0) + \Big(\max\{0, \beta_0\} + 10d \Big)^2. \tag{ReLU}$$

Observe that solving (ReLU) is equivalent to training a ReLU-regression where the data samples are:

1. $\boldsymbol{X}_1 = [a_1, \ldots, a_d]$, $Y_1 = \frac{1}{2} \sum_{i=1}^{d} a_i$
2. $\boldsymbol{X}_2 = [2 \cdot a_1, \ldots, 2 \cdot a_d]$, $Y_1 = \sum_{i=1}^{d} a_i$
3. $\boldsymbol{X}_{2i+1} = \boldsymbol{e}_i$, $Y_{2i+1} = 1$, $\boldsymbol{X}_{2i+2} = -\boldsymbol{e}_i$, $Y_{2i+2} = 1$ for $i \in \{1, \ldots, d\}$.
4. $\boldsymbol{X}_{2d+3} = \boldsymbol{0}$, $Y_{2d+3} = 10d$.

Now we verify Theorem 8 by showing that the $\{\pm 1\}-$subset sum problem iff the training error in solving (ReLU) is $d + 100d^2$.

Thus

- Suppose the $\{\pm 1\}-$subset sum problem with non-negative parameters $a_1, \ldots, a_d$

has a feasible solution $x \in \{\pm 1\}^d$ such that $\sum_{i=1}^d a_i x_i = \frac{1}{2}\sum_{i=1}^d a_i$. Let $\boldsymbol{\beta} = x$ and $\beta_0 = 0$, we have that the objective function value of (ReLU) is $d + 100d^2$.

- Suppose the $\{\pm 1\}-$subset sum problem does not have a feasible solution. Let $\boldsymbol{\beta}, \beta_0$ be the optimal solution to (ReLU). Then, observe that

$$g(\beta_0) = \begin{cases} 2\beta_0^2 - 4\beta_0 + 2 & \text{if } \beta_0 \geq 1 - \frac{\sqrt{2}}{2} \\ 1 & \text{if } \beta_0 \leq 1 - \frac{\sqrt{2}}{2} \ (\leq \frac{1}{2}) \end{cases}$$

We consider four cases:

1. $\beta_0 \geq 1 - \frac{\sqrt{2}}{2}$: In this case

$$\begin{aligned} & \tau(\boldsymbol{\beta}, \beta_0) + d \cdot g(\beta_0) + (\max\{0, \beta_0\} + 10d)^2 \\ \geq\ & 0 + d(2\beta_0^2 - 4\beta_0 + 2) + (\beta_0 + 10d)^2 \\ =\ & (2d+1)\beta_0^2 + 16d\beta_0 + 2d + 100d^2 \\ >\ & d + 100d^2. \end{aligned}$$

2. $0 < \beta_0 \leq 1 - \frac{\sqrt{2}}{2}$: In this case

$$\begin{aligned} & \tau(\boldsymbol{\beta}, \beta_0) + d \cdot g(\beta_0) + (\max\{0, \beta_0\} + 10d)^2 \\ \geq\ & 0 + d \times 1 + (\beta_0 + 10d)^2 \\ >\ & d + 100d^2 \end{aligned}$$

3. $\beta_0 < 0$: Note that $\frac{1}{2}\sum_{i=1}^d a_i > 0$ and therefore $\tau(\boldsymbol{\beta}, \beta_0) = 0$ iff $\beta_0 = 0$. In particular in this case $\tau(\boldsymbol{\beta}, \beta_0) > 0$. Therefore, we have $\tau(\boldsymbol{\beta}, \beta_0) + d \cdot g(\beta_0) + (\max\{0, \beta_0\} + 10d)^2 > 0 + d \cdot 1 + 100d^2$.

4. $\beta_0 = 0$: In this case, observe that the objective function value $\tau(\boldsymbol{\beta}, \beta_0) + \sum_{i=1}^d \theta(\boldsymbol{e}_j^\top \boldsymbol{\beta}, \beta_0) + (\max\{0, \beta_0\} + 10d)^2$ is greater than $0 + d \cdot 1 + 100d^2$. How-

ever, for equality to hold in the above inequality, we must have $\theta(\boldsymbol{e}_j^\top \boldsymbol{\beta}, \beta_0) = 1$ for $j \in [d]$ and $\tau(\boldsymbol{\beta}, \beta_0) = 0$, which implies we must have $\boldsymbol{\beta}_j \in \{-1, 1\}$ and $\sum_{i=1}^d a_i \boldsymbol{\beta}_i = \frac{1}{2} \sum_{i=1}^d a_i$. However since there is no solution to the $\{\pm 1\}-$subset sum problem, we obtain that $\tau(\boldsymbol{\beta}, \beta_0) + \sum_{i=1}^d \theta(\boldsymbol{e}_j^\top \boldsymbol{\beta}, \beta_0) + (\max\{0, \beta_0\} + 10d)^2 > 0 + d \cdot 1 + 100d^2$.

C.3 Proof of Proposition 3.2.2

Proof. Rewrite

$$\| \max\{\mathbf{0}, \boldsymbol{X}\boldsymbol{\beta} + \beta_0 \mathbf{1}\} - Y\|_2^2 = \sum_{i=1}^m (\max\{0, \boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0\} - Y_i)^2 + \phi(\boldsymbol{\beta}, \beta_0).$$

Note that $Y_i > 0$ for all $i \in [m]$, then we have:

$$(\max\{0, \boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0\} - Y_i)^2 \leq (\boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0 - Y_i)^2$$
$$(\max\{0, \boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0\} - Y_i)^2 \leq \sigma(\boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0, Y_i)$$

holds for all $i \in [m]$. Since for any $I \subseteq [m]$

$$\begin{aligned} &\sum_{i=1}^m (\max\{0, \boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0\} - Y_i)^2 + \phi(\boldsymbol{\beta}, \beta_0) \\ \leq &\sum_{i \in I} (\boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0 - Y_i)^2 + \sum_{i \in [m] \setminus I} \sigma(\boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0, Y_i) + \phi(\boldsymbol{\beta}, \beta_0), \end{aligned}$$

then taking minimum on both side implies $\min_{(\boldsymbol{\beta}, \beta_0) \in \mathbb{R}^p \times \mathbb{R}} \| \max\{\mathbf{0}, \boldsymbol{X}\boldsymbol{\beta} + \beta_0 \mathbf{1}\} - Y\|_2^2 \leq z^\sigma(I)$.

Moreover, recall I^{opt} is the active set corresponding to a globally optimal solution

$(\beta^{\text{opt}}, \beta_0^{\text{opt}})$ as defined above. We have:

$$\begin{aligned} z^\sigma(I^{\text{opt}}) \leq & \sum_{i \in I^{\text{opt}}} (\underbrace{\boldsymbol{X}_i^\top \boldsymbol{\beta}^{\text{opt}} + \beta_0^{\text{opt}}}_{\geq 0} - Y_i)^2 + \sum_{i \in [m] \backslash I^{\text{opt}}} \sigma(\underbrace{\boldsymbol{X}_i^\top \boldsymbol{\beta}^{\text{opt}} + \beta_0^{\text{opt}}}_{<0}, Y_i) + \phi(\boldsymbol{\beta}^{\text{opt}}, \beta_0^{\text{opt}}) \\ = & \sum_{i \in I^{\text{opt}}} (\boldsymbol{X}_i^\top \boldsymbol{\beta}^{\text{opt}} + \beta_0^{\text{opt}} - Y_i)^2 + \sum_{i \in [m] \backslash I^{\text{opt}}} Y_i^2 + \phi(\boldsymbol{\beta}^{\text{opt}}, \beta_0^{\text{opt}}) \\ = & \, z^{\text{opt}}. \end{aligned}$$

Combine with $z^{\text{opt}} = \| \max\{\mathbf{0}, \boldsymbol{X}\boldsymbol{\beta} + \beta_0 \mathbf{1}\} - Y \|_2^2 \leq z^\sigma(I^{\text{opt}})$, we have $z^{\text{opt}} = z^\sigma(I^{\text{opt}})$. □

C.4 Proof of Proposition 3.2.2

Proof. Recall that $(\boldsymbol{\beta}^{\text{opt}}, \beta_0^{\text{opt}})$ is a globally optimal solution, and z^{opt} is the globally optimal value of ReLU-regression. Let $I^{\text{opt}} = \left\{ i : \boldsymbol{X}_i^\top \boldsymbol{\beta}^{\text{opt}} + \beta_0^{\text{opt}} > 0 \right\} \subseteq [m]$ be the active set corresponds to $(\boldsymbol{\beta}^{\text{opt}}, \beta_0^{\text{opt}})$. Based on the input condition of Algorithm 4, the response samples $\{Y_i\}$ satisfies:

$$0 < Y_1 \leq Y_2 \leq \ldots \leq Y_m.$$

Given k as a predefined integral parameter, pick k indices $i_1, i_2, \ldots, i_k$ such that $0 \leq i_1 < \ldots < i_k \leq m$, from Algorithm 4, let

$$[m] \backslash \hat{I} := \{1, \ldots, i_1\} \cup \left(\bigcup_{\ell=2}^{k} \{i_\ell\} \right)$$

be the inactive set, and

$$\hat{I} := \left(\bigcup_{\ell=1}^{k-1} \{i_\ell + 1, \ldots, i_{\ell+1} - 1\} \right) \cup \{i_k + 1, \ldots, m\}$$

be the active set.

Suppose I^{opt} is of size $|I^{\text{opt}}| \geq m - k + 1$, let $\{s_\ell\}_{\ell=1}^d$ with $d \leq k - 1$ be the set of

increasingly-sorted indices that are not in I^{opt}. Let $j = d + 1 \leq k$, set $i_1 = 0$, $i_\ell = s_{\ell-1}$, for all $\ell = 2, \ldots, j$. Then we see that Algorithm 4 would discover the optimal solution and thus solve the ReLU-regression problem exactly. Therefore, henceforth we assume that $|I^{\text{opt}}| \leq m - k$.

Now pick $i_1, \ldots, i_k$ as the largest increasingly-sorted indices that not in I^{opt}. Therefore we have: (1) $\hat{I} \subseteq I^{\text{opt}}$, (2) $\bigcup_{\ell=1}^{k} \{i_\ell\} \subseteq [m] \backslash I^{\text{opt}}$, and (3) $i_k - 1 \in I^{\text{opt}}$ if $i_{k-1} \neq i_k - 1$, these three conditions further implies that

$$I^{\text{opt}} \backslash \hat{I} \subseteq \{1, \ldots, i_1 - 1\}.$$

Since the approximation algorithm examines this solution, we will use this "solution" to obtain an upper bound on the quality of solution produced by the Algorithm.

Thus the objective value $z^\sigma(\hat{I})$ is further upper bounded as follows:

$$\begin{aligned}
z^\sigma(\hat{I}) = & \min_{(\boldsymbol{\beta}, \beta_0) \in \mathbb{R}^d \times \mathbb{R}} \sum_{i \in \hat{I}} (\boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0 - Y_i)^2 + \sum_{i \in [m] \backslash \hat{I}} \sigma(\boldsymbol{X}_i^\top \boldsymbol{\beta} + \beta_0, Y_i) + \phi(\boldsymbol{\beta}, \beta_0) \\
\leq & \sum_{i \in \hat{I}} (\underbrace{\boldsymbol{X}_i^\top \boldsymbol{\beta}^{\text{opt}} + \beta_0^{\text{opt}}}_{\geq 0} - Y_i)^2 + \sum_{i \in I^{\text{opt}} \backslash \hat{I}} \sigma(\underbrace{\boldsymbol{X}_i^\top \boldsymbol{\beta}^{\text{opt}} + \beta_0^{\text{opt}}}_{\geq 0}, Y_i) \\
& + \sum_{i \in [m] \backslash I^{\text{opt}}} \sigma(\underbrace{\boldsymbol{X}_i^\top \beta^{\text{opt}} + \beta_0^{\text{opt}}}_{< 0}, Y_i) + \phi(\boldsymbol{\beta}^{\text{opt}}, \beta_0^{\text{opt}}) \\
= & \sum_{i \in \hat{I}} (\boldsymbol{X}_i^\top \boldsymbol{\beta}^{\text{opt}} + \beta_0^{\text{opt}} - Y_i)^2 + \sum_{i \in I^{\text{opt}} \backslash \hat{I}} \sigma(\boldsymbol{X}_i^\top \boldsymbol{\beta}^{\text{opt}} + \beta_0^{\text{opt}}, Y_i) \\
& + \sum_{i \in [m] \backslash I^{\text{opt}}} Y_i^2 + \phi(\boldsymbol{\beta}^{\text{opt}}, \beta_0^{\text{opt}}).
\end{aligned}$$

Split $I^{\text{opt}} \backslash \hat{I}$ into the following two parts:

$$\begin{aligned}
\tilde{I}_+ &:= \left\{ i \in I^{\text{opt}} \backslash \hat{I} : \boldsymbol{X}_i^\top \boldsymbol{\beta}^{\text{opt}} + \beta_0^{\text{opt}} > 2Y_i \right\}, \\
\tilde{I}_- &:= \left\{ i \in I^{\text{opt}} \backslash \hat{I} : 2Y_i \geq \boldsymbol{X}_i^\top \boldsymbol{\beta}^{\text{opt}} + \beta_0^{\text{opt}} \geq 0 \right\},
\end{aligned}$$

the second term of above equals to:

$$\sum_{i\in I^{\text{opt}}\setminus\hat{I}} \sigma(\boldsymbol{X}_i^\top \boldsymbol{\beta}^{\text{opt}} + \beta_0^{\text{opt}}, Y_i) = \sum_{i\in\tilde{I}_+} (\boldsymbol{X}_i^\top \boldsymbol{\beta}^{\text{opt}} + \beta_0^{\text{opt}} - Y_i)^2 + \sum_{i\in\tilde{I}_-} Y_i^2.$$

Therefore,

$$z^\sigma(\hat{I}) \leq \sum_{i\in\hat{I}\cup\tilde{I}_+} (\boldsymbol{X}_i^\top \boldsymbol{\beta}^{\text{opt}} + \beta_0^{\text{opt}} - Y_i)^2 + \sum_{i\in\tilde{I}_-\cup([m]\setminus I^{\text{opt}})} Y_i^2 + \phi(\boldsymbol{\beta}^{\text{opt}}, \beta_0^{\text{opt}}). \tag{UB}$$

Since $I^{\text{opt}} = \hat{I} \cup \tilde{I}_+ \cup \tilde{I}_-$, then the globally optimal value of ReLU-regression can be represented as

$$\begin{aligned}
& z^{\text{opt}} \\
&= \sum_{i\in I^{\text{opt}}} (\boldsymbol{X}_i^\top \boldsymbol{\beta}^{\text{opt}} + \beta_0^{\text{opt}} - Y_i)^2 + \sum_{i\in[m]\setminus I^{\text{opt}}} Y_i^2 + \phi(\boldsymbol{\beta}^{\text{opt}}, \beta_0^{\text{opt}}) \\
&= \sum_{i\in\hat{I}\cup\tilde{I}_+} (\boldsymbol{X}_i^\top \boldsymbol{\beta}^{\text{opt}} + \beta_0^{\text{opt}} - Y_i)^2 + \sum_{i\in\tilde{I}_-} (\boldsymbol{X}_i^\top \boldsymbol{\beta}^{\text{opt}} + \beta_0^{\text{opt}} - Y_i)^2 + \sum_{i\in[m]\setminus I^{\text{opt}}} Y_i^2 + \phi(\boldsymbol{\beta}^{\text{opt}}, \beta_0^{\text{opt}}).
\end{aligned}$$

Note $\{i_1, \ldots, i_k\}$ is a subset of $[m]\setminus I^{\text{opt}}$ based on our choice $i_1, \ldots, i_k$, then define the term D as:

$$D := \underbrace{\sum_{i\in\hat{I}\cup\tilde{I}_+} (\boldsymbol{X}_i^\top \boldsymbol{\beta}^{\text{opt}} + \beta_0^{\text{opt}} - Y_i)^2 + \sum_{i\in[m]\setminus I^{\text{opt}}} Y_i^2 + \phi(\boldsymbol{\beta}^{\text{opt}}, \beta_0^{\text{opt}})}_{\geq \sum_{j=1}^k Y_{i_j}^2}. \tag{D-ineq}$$

Since UB and z^{opt} can be represented as:

$$(\text{UB}) := D + \sum_{i\in\tilde{I}_-} Y_i^2, \qquad z^{\text{opt}} := D + \sum_{i\in\tilde{I}_-} (\boldsymbol{X}_i^\top \boldsymbol{\beta}^{\text{opt}} + \beta_0^{\text{opt}} - Y_i)^2$$

then the approximation ratio ρ guaranteed by Algorithm 4 is upper bounded as follows:

$$\rho := \frac{z^{\sigma}(\hat{I})}{z^{\text{opt}}} \leq \frac{(UB)}{z^{\text{opt}}} = \frac{D + \sum_{i \in \tilde{I}_-} Y_i^2}{D + \sum_{i \in \tilde{I}_-} (X_i^\top \boldsymbol{\beta}^{\text{opt}} + \beta_0^{\text{opt}} - Y_i)^2} \leq \frac{D + \sum_{i \in \tilde{I}_-} Y_i^2}{D} \leq \frac{n}{k}$$

where the final inequality holds because of the following: with $\{Y_i\}_{i=1}^m$ increasingly-sorted, the term $\sum_{i \in \tilde{I}_-} Y_i^2$ can be upper bounded by

$$\begin{array}{rcll} \sum_{i \in \tilde{I}_-} Y_i^2 & \leq & |\tilde{I}_-| \cdot Y_{i_1}^2 & \text{by } \tilde{I}_- \subseteq \{1, \ldots, i_1 - 1\}, \\ & \leq & \frac{|\tilde{I}_-|}{k} \cdot \sum_{j=1}^k Y_{i_j}^2 & \text{by } Y_{i_j} \geq Y_{i_1} \text{ for all } j \in [k], \\ & \leq & \frac{|\tilde{I}_-|}{k} \cdot D & \text{by previous inequality of } D, \\ & \leq & \frac{n-k}{k} \cdot D & \text{by } \tilde{I}_- \subseteq I^{\text{opt}},\ |I^{\text{opt}}| \leq n - k, \end{array}$$

then replacing $\sum_{i \in \tilde{I}_-} Y_i^2$ by $\frac{n-k}{k} \cdot D$ gives the final approximation ratio. □

C.5 Proof of Theorem 10

We first verify Proposition 3.2.3 and Proposition 3.2.4. Proposition 3.2.3 is a consequence of the following result:

Theorem 11 (Mickey, 1963)**.** *Let g be a function on $\mathcal{X} \times \Theta$ where $\mathcal{X}$ is a Euclidean space and Θ is a compact set of Euclidean space. Let $g(x, \theta)$ be a continuous function of θ for each $x \in \mathcal{X}$ and a measurable function of x for each θ. Assume assume that $|g(x, \theta)| \leq h(x)$ for all $x \in \mathcal{X}$ and $\theta \in \Theta$, where h is integrable with respect to a probability distribution function F on $\mathcal{X}$. If $x_1, x_2, \ldots$ is a random sample from F then for almost every sequence $\{x_t\}$*

$$\frac{1}{n} \sum_{t=1}^{n} g(x_t, \theta) \to \int g(x, \theta) dF(x)$$

uniformly for all $\theta \in \Theta$.

Proof. of Proposition 3.2.3 Let $\psi_y(\boldsymbol{X}\boldsymbol{\beta} + \beta_0, Y)$ be defined as in Proposition 3.2.3. Let

$\mathcal{X} = \mathbb{R}^p \times \mathbb{R}$ be a Euclidean space, and let Θ be the same convex compact set in Assumption 3.2.1. We have $\psi_y(\boldsymbol{X}\boldsymbol{\beta} + \beta_0, Y)$ is a continuous function of $(\boldsymbol{\beta}, \beta_0)$ for each $(\boldsymbol{X}, Y) \in \mathcal{X}$ and a measurable function of $(\boldsymbol{X}, Y)$ for each $(\boldsymbol{\beta}, \beta_0) \in \Theta$. Moreover, since Θ is a convex compact set, then there exists a constant $d_\Theta > 0$ such that $|\theta_i| \leq d_\Theta$ for all $i = 0, 1, \ldots, d$. Define function $h(\boldsymbol{X}, Y)$ as

$$h(\boldsymbol{X}, Y) = 2\left(\sum_{i=1}^{d} |[\boldsymbol{X}]_i| \cdot d_\Theta + d_\Theta\right)^2 + 2Y^2$$

where $[\boldsymbol{X}]_i$ denotes the i^{th} component of $\boldsymbol{X}$ for $i = 1, \ldots, d$. Thus we have $h(\boldsymbol{X}, Y) \geq |\psi_y(\boldsymbol{X}\boldsymbol{\beta} + \beta_0, Y)|$ holds for all $(\boldsymbol{X}, Y) \in \mathcal{X}$ and $(\boldsymbol{\beta}, \beta_0) \in \Theta$, where $h(\boldsymbol{X}, Y)$ is integrable with respect to a probability distribution $\mathcal{N} \times \mathcal{D}$ on $\mathcal{X}$. Since all the conditions in Theorem 11 holds, Proposition 3.2.3 holds. □

Proposition 3.2.4 is a consequence of the following result:

Theorem 12 (Jennrich, 1969)**.** *Under the statistical model: $y_t = f(x_t, \theta_0) + \epsilon_t$ for all $t = 1, \ldots, n$ when x_t is i^{th} "fixed" input vector and $\{\epsilon_t\}$ are i.i.d. distributed errors with zero mean and same finite unknown variance. Any vector $\hat{\theta}_n \in \Theta$ which minimizes the residual sum of squares*

$$S_n(\theta) := \frac{1}{n}\sum_{t=1}^{n}(f(x_t, \theta) - y_t)^2$$

is said to be strongly consistent of θ_0 (i.e., $\hat{\theta}_n \to \theta_0$ almost surely as $n \to \infty$) under the following condition: $D_n(\theta, \theta')$ convergence uniformly to a continuous function $D(\theta, \theta')$ and $D(\theta, \theta_0) = 0$ if and only if $\theta = \theta_0$ where

$$D_n(\theta, \theta') = \frac{1}{n}\sum_{t=1}^{n}(f(x_t, \theta) - f(x_t, \theta'))^2.$$

Proof. of Proposition 3.2.4 Based on Theorem 11, with the similar proof of Proposi-

tion 3.2.3, we have:

$$\underbrace{\frac{1}{n}\sum_{i=1}^{n}\Big(\max\{0, \boldsymbol{X}_i^\top\boldsymbol{\beta}+\beta_0\}-\max\{0,\boldsymbol{X}_i^\top\boldsymbol{\beta}^*+\beta_0^*\}\Big)^2}_{=:D_n((\boldsymbol{\beta},\beta_0),(\boldsymbol{\beta}^*,\beta_0^*))}$$
$$\to \underbrace{\mathbb{E}_{\boldsymbol{X}\sim\mathcal{N},\epsilon\sim\mathcal{D}}\left[\Big(\max\{0, \boldsymbol{X}^\top\boldsymbol{\beta}+\beta_0\}-\max\{0,\boldsymbol{X}^\top\boldsymbol{\beta}^*+\beta_0^*\}\Big)^2\right]}_{=:D((\boldsymbol{\beta},\beta_0),(\boldsymbol{\beta}^*,\beta_0^*))}$$

uniformly for almost every sequence $\{\boldsymbol{X}_i, Y_i\}$. Moreover, a direct consequence of the second property of distribution $\mathcal{N}$ (Unique Optimal Property) implies that $D((\boldsymbol{\beta},\beta_0),(\boldsymbol{\beta}^*,\beta_0^*)) = 0$ if and only if $(\boldsymbol{\beta},\beta_0)=(\boldsymbol{\beta}^*,\beta_0^*)$. Thus, since all conditions of Theorem 12 hold, Proposition 3.2.4 holds. □

Proof. of Theorem 10 The optimal value of the asymptotic objective function from sorting algorithm can be upper bounded by replacing optimal solution with the true parameter $\boldsymbol{\beta}^*$ as follows:

$$\min_{\boldsymbol{\beta}\in\Theta}\mathbb{E}_{\boldsymbol{X}\sim\mathcal{N},\epsilon\sim\mathcal{D}}[\psi_y(\boldsymbol{X}^\top\boldsymbol{\beta}+\beta_0,Y)] \le \mathbb{E}_{\boldsymbol{X}\sim\mathcal{N},\epsilon\sim\mathcal{D}}[\psi_y(\boldsymbol{X}^\top\boldsymbol{\beta}^*+\beta^*,Y)],$$

where $\mathbb{E}_{\boldsymbol{X}\sim\mathcal{N},\epsilon\sim\mathcal{D}}[\psi_y(\boldsymbol{X}^\top\boldsymbol{\beta}^*+\beta_0^*,Y)]$ can be split into the sum from (T_1) to (T_7):

$$\begin{aligned}
&\mathbb{E}_{X\sim\mathcal{N},\epsilon\sim\mathcal{D}}\big[\psi_y(\boldsymbol{X}^\top\boldsymbol{\beta}^*+\beta_0^*,Y)\big]\\
&=\mathbb{E}\big[Y^2 \mid 0<Y\le y, 0<\boldsymbol{X}^\top\boldsymbol{\beta}^*+\beta_0^*\le 2Y\big]\cdot\mathbb{P}(0<Y\le y, 0<\boldsymbol{X}^\top\boldsymbol{\beta}^*+\beta_0^*\le 2Y) && (T_1)\\
&\quad+\mathbb{E}\big[Y^2 \mid 0<Y\le y, \boldsymbol{X}^\top\boldsymbol{\beta}^*+\beta_0^*\le 0\big]\cdot\mathbb{P}(0<Y\le y, \boldsymbol{X}^\top\boldsymbol{\beta}^*+\beta_0^*\le 0) && (T_2)\\
&\quad+\mathbb{E}\big[(\boldsymbol{X}^\top\boldsymbol{\beta}^*+\beta_0^*-Y)^2 \mid 0<Y\le y, 2Y<\boldsymbol{X}^\top\boldsymbol{\beta}^*+\beta_0^*\big]\cdot\mathbb{P}(0<Y\le y, 2Y<\boldsymbol{X}^\top\boldsymbol{\beta}^*+\beta_0^*) && (T_3)\\
&\quad+\mathbb{E}\big[(\boldsymbol{X}^\top\boldsymbol{\beta}^*+\beta_0^*-Y)^2 \mid y<Y, 0<\boldsymbol{X}^\top\boldsymbol{\beta}^*+\beta_0^*\big]\cdot\mathbb{P}(y<Y, 0<\boldsymbol{X}^\top\boldsymbol{\beta}^*+\beta_0^*) && (T_4)\\
&\quad+\mathbb{E}\big[(\boldsymbol{X}^\top\boldsymbol{\beta}^*+\beta_0^*-Y)^2 \mid y<Y, \boldsymbol{X}^\top\boldsymbol{\beta}^*+\beta_0^*\le 0\big]\cdot\mathbb{P}(y<Y, \boldsymbol{X}^\top\boldsymbol{\beta}^*+\beta_0^*\le 0) && (T_5)\\
&\quad+\mathbb{E}\big[\epsilon^2 \mid Y\le 0, \boldsymbol{X}^\top\boldsymbol{\beta}^*+\beta_0^*\le 0\big]\cdot\mathbb{P}(Y\le 0, \boldsymbol{X}^\top\boldsymbol{\beta}^*+\beta_0^*\le 0) && (T_6)\\
&\quad+\mathbb{E}\big[\epsilon^2 \mid Y\le 0, 0<\boldsymbol{X}^\top\boldsymbol{\beta}^*+\beta_0^*\big]\cdot\mathbb{P}(Y\le 0, 0<\boldsymbol{X}^\top\boldsymbol{\beta}^*+\beta_0^*). && (T_7)
\end{aligned}$$

Since term (T_1) - (T_7) can be reformulated as follows:

$$
\begin{aligned}
(T_1) &= \mathbb{E}[Y^2 \mid 0 < Y \le y, 0 < \boldsymbol{X}^\top \boldsymbol{\beta}^* + \beta_0^* \le 2Y] \cdot \mathbb{P}(0 < Y \le y, 0 < \boldsymbol{X}^\top \boldsymbol{\beta}^* + \beta_0^* \le 2Y), \\
(T_2) &= \mathbb{E}[\epsilon^2 \mid 0 < Y \le y, \boldsymbol{X}^\top \boldsymbol{\beta}^* + \beta_0^* \le 0] \cdot \mathbb{P}(0 < Y \le y, \boldsymbol{X}^\top \boldsymbol{\beta}^* + \beta_0^* \le 0), \\
(T_3) &= \mathbb{E}[\epsilon^2 \mid 0 < Y \le y, 2Y < \boldsymbol{X}^\top \boldsymbol{\beta}^* + \beta_0^*] \cdot \mathbb{P}(0 < Y \le y, 2Y < \boldsymbol{X}^\top \boldsymbol{\beta}^* + \beta_0^*), \\
(T_4) &= \mathbb{E}[\epsilon^2 \mid y < Y, 0 < \boldsymbol{X}^\top \boldsymbol{\beta}^* + \beta_0^*] \cdot \mathbb{P}(y < Y, 0 < \boldsymbol{X}^\top \boldsymbol{\beta}^* + \beta_0^*), \\
(T_5) &= \mathbb{E}\big[(\boldsymbol{X}^\top \boldsymbol{\beta}^* + \beta_0^* - \epsilon)^2 \mid y < \epsilon, \boldsymbol{X}^\top \boldsymbol{\beta}^* + \beta_0^* \le 0\big] \cdot \mathbb{P}(y < \epsilon, \boldsymbol{X}^\top \boldsymbol{\beta}^* + \beta_0^* \le 0), \\
(T_6) &= \mathbb{E}\big[\epsilon^2 \mid Y \le 0, \boldsymbol{X}^\top \boldsymbol{\beta}^* + \beta_0^* \le 0\big] \cdot \mathbb{P}(Y \le 0, \boldsymbol{X}^\top \boldsymbol{\beta}^* + \beta_0^* \le 0), \\
(T_7) &= \mathbb{E}\big[\epsilon^2 \mid Y \le 0, 0 < \boldsymbol{X}^\top \boldsymbol{\beta}^* + \beta_0^*\big] \cdot \mathbb{P}(Y \le 0, 0 < \boldsymbol{X}^\top \boldsymbol{\beta}^* + \beta_0^*),
\end{aligned}
$$

note that (T_1) is upper bounded by y^2, $(T_2) + (T_3) + (T_4) + (T_6) + (T_7) \le \text{Var}(\epsilon) = \gamma^2$, and by Lemma C.5.1 (proved below) and setting $y = 0$,

$$
z^{\text{asy}} \le \mathbb{E}_{\boldsymbol{X} \sim \mathcal{N}, \epsilon \sim \mathcal{D}}[\psi_0(\boldsymbol{X}^\top \boldsymbol{\beta}^* + \beta_0^*, Y)] \le \gamma^2 + \frac{\gamma^2}{2} + \frac{2 + 2\Delta^2}{\sqrt{2\pi}}\gamma.
$$

To lower bound z^{asy}, note that $\psi_y(\boldsymbol{X}^\top \boldsymbol{\beta}^* + \beta_0, Y) \ge \big(\max\{0, \boldsymbol{X}^\top \boldsymbol{\beta}^* + \beta_0\} - Y\big)^2$ holds for any $(\boldsymbol{\beta}, \beta_0) \in \Theta$ and any $(\boldsymbol{X}, Y) \in \mathcal{X}$, and by Proposition 3.2.4, the optimal value of asymptotic version of ReLU-regression problem is γ^2, thus z^* is lower bounded by γ^2. Combine lower and upper bounds together, we have

$$
\gamma^2 \le z^{\text{asy}} \le \frac{3\gamma^2}{2} + \frac{2 + 2\Delta^2}{\sqrt{2\pi}}\gamma.
$$

□

Lemma C.5.1. *Assume the underlying statistical model 3.2.1 holds, we have*

$$
(T_5) \le \frac{\gamma^2}{2} + \frac{2 + 2\Delta^2}{\sqrt{2\pi}}\gamma.
$$

Proof. Assume the underlying statistical model 3.2.1, we have $\boldsymbol{X}^\top \boldsymbol{\beta}^* + \beta_0^*$ satisfies $\mathbb{E}[\boldsymbol{X}^\top \boldsymbol{\beta}^* +$

$\beta_0^*] = \beta_0^*, \text{Var}(\boldsymbol{X}^\top\boldsymbol{\beta}^* + \beta_0^*) = (\boldsymbol{\beta}^*)^\top\boldsymbol{\Sigma}\boldsymbol{\beta}^* = \Delta^2$, thus (T_5) is upper bounded by

$$\begin{aligned}
&\mathbb{E}\big[(\boldsymbol{X}^\top\boldsymbol{\beta}^* + \beta_0^* - \epsilon)^2 \mid y < \epsilon, \boldsymbol{X}^\top\boldsymbol{\beta}^* + \beta_0^* \le 0\big] \cdot \mathbb{P}(y < \epsilon, \boldsymbol{X}^\top\boldsymbol{\beta}^* + \beta_0^* \le 0) \\
= &\int_{\substack{v\in\mathbb{R}\\ u\in\mathbb{R}}} (u-v)^2 f \mathrm{d}u\mathrm{d}v \cdot \mathbb{P}(y < \epsilon, \boldsymbol{X}^\top\boldsymbol{\beta}^* + \beta_0^* \le 0) \qquad (*)
\end{aligned}$$

where $f := f(\epsilon = v, \boldsymbol{X}^\top\boldsymbol{\beta}^* + \beta_0^* = u \,|y < \epsilon, \boldsymbol{X}^\top\boldsymbol{\beta}^* + \beta_0^* \le 0)$ is the conditional joint density function of variables $\epsilon, \boldsymbol{X}^\top\boldsymbol{\beta}^* + \beta_0^*$. Then

$$\begin{aligned}
(*) = &\int_{\substack{v>y\\ u\le 0}} (u^2 - 2uv + v^2) f(\epsilon = v) f(\boldsymbol{X}^\top\boldsymbol{\beta}^* + \beta_0^* = u)\mathrm{d}u\mathrm{d}v \\
= &\int_{v>y} f(\epsilon = v)\mathrm{d}v \cdot \int_{u\le 0} u^2 f(\boldsymbol{X}^\top\boldsymbol{\beta}^* + \beta_0^* = u)\mathrm{d}u \\
&- 2\int_{v>y} v f(\epsilon = v)\mathrm{d}v \cdot \int_{u\le 0} u f(\boldsymbol{X}^\top\boldsymbol{\beta}^* + \beta_0^* = u)\mathrm{d}u \\
&+ \int_{v>y} v^2 f(\epsilon = v)\mathrm{d}v \cdot \int_{u\le 0} f(\boldsymbol{X}^\top\boldsymbol{\beta}^* + \beta_0^* = u)\mathrm{d}u
\end{aligned}$$

where

$$\begin{aligned}
&\int_{u\le 0} u^2 f(\boldsymbol{X}^\top\boldsymbol{\beta}^* + \beta_0^* = u)\mathrm{d}u &&\le \Delta^2, \\
-1 - \Delta^2 \le &\int_{u\le 0} u f(\boldsymbol{X}^\top\boldsymbol{\beta}^* + \beta_0^* = u)\mathrm{d}u &&\le 1 + \Delta^2, \\
&\int_{u\le 0} f(\boldsymbol{X}^\top\boldsymbol{\beta}^* + \beta_0^* = u)\mathrm{d}u &&\le 1.
\end{aligned}$$

Suppose the noise ϵ follows Gaussian distribution $N(0, \gamma^2)$, then

$$\begin{aligned}
(*) &\le \int_{v>y} f(\epsilon = v)\mathrm{d}v \cdot \Delta^2 + 2\int_{v>y} v f(\epsilon = v)\mathrm{d}v \cdot \big(1 + \Delta^2\big) + \int_{v>y} v^2 f(\epsilon = v)\mathrm{d}v \cdot 1 \\
&= \frac{1}{2}\text{erf}\left(\frac{y}{\sqrt{2}\gamma}\right) \cdot \Delta^2 + \frac{2}{\sqrt{2\pi}} e^{-\frac{y^2}{2\gamma^2}}\gamma \cdot \big(1 + \Delta^2\big) + \left(\frac{1}{\sqrt{2\pi}} y\gamma e^{-\frac{y^2}{2\gamma^2}} + \frac{\gamma^2}{2}\text{erfc}\left(\frac{y}{\sqrt{2}\gamma}\right)\right) \cdot 1 \\
&\le \frac{\Delta^2 y}{\sqrt{2\pi}\gamma} + \frac{2 + 2\Delta^2}{\sqrt{2\pi}}\gamma e^{-\frac{y^2}{2\gamma^2}} + \frac{1}{\sqrt{2\pi}} y\gamma e^{-\frac{y^2}{2\gamma^2}} + \frac{\gamma^2}{2} e^{-\frac{y^2}{2\gamma^2}}
\end{aligned}$$

where the final inequality holds since

$$\text{erf}(z) := \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt \leq \frac{2z}{\sqrt{\pi}}, \qquad \text{erfc}(z) := 1 - \text{erf}(z) \leq e^{-z^2}.$$

Since the above inequality holds for any $y \geq 0$, then set $y = 0$, we have

$$(*) \leq \frac{\gamma^2}{2} + \frac{2 + 2\Delta^2}{\sqrt{2\pi}}\gamma.$$

□

C.6 Methods for Comparison in Numerical Experiments

C.6.1 Sorting Method

The sorting method that used in Section 3.3.2 is a special case of Algorithm 4 which follows Algorithm 8. The above sorting method is a special case of Algorithm 4 with parameter

Algorithm 8 Sorting Method

Input: Set of sample points $\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n \in \mathbb{R}^d \times \mathbb{R}$, integer $1 \leq T \leq n$.
Output: A feasible solution $\hat{\boldsymbol{\beta}}$.

1: Without loss of generality, sort $\{Y_i\}_{i=1}^n$ as $Y_1 \leq Y_2 \leq \ldots \leq Y_n$.
2: **for** $t = 0, 1, \ldots, T$ **do**
3: Set $\mathcal{I}^t \leftarrow \{\lfloor \frac{t}{N} n \rfloor + 1, \ldots, N\} \subseteq \{1, \ldots, T\}$ for $t = 0, 1, \ldots, T-1$, and $\mathcal{I}^T \leftarrow \emptyset$.
4: Set $\boldsymbol{\beta}^t \leftarrow \arg\min_{\beta \in \mathbb{R}^p} f^{\sigma}_{\mathcal{I}^t}(\boldsymbol{\beta})$.
5: Compute the objective value of the ReLU-regression with $\boldsymbol{\beta}^t$ as

$$\text{opt}^t \leftarrow \sum_{i=1}^n (\max\{0, \boldsymbol{X}_i^\top \boldsymbol{\beta}^t\} - Y_i)^2.$$

6: **end for**
7: **return** $\hat{\boldsymbol{\beta}}$ where $\hat{\boldsymbol{\beta}}$ is a feasible solution with the minimum opt^t.

$k = 1$ and subset

$$\{i\} = \begin{cases} \{\lfloor \frac{t}{T} n \rfloor\} & \text{if } t = 1, \ldots, T \\ \emptyset & \text{if } t = 0 \end{cases}$$

which implies the term corresponds to $\lfloor \frac{t}{T} n \rfloor$-th index in the objective function of ReLU-regression is not in the quadratic part (i.e., not active) but in the σ function part.

C.6.2 Appendix: Iterative Method

Given any feasible solution $\boldsymbol{\beta}$ of the ReLU-regression problem, let the *iterative set* $\mathcal{I}(\boldsymbol{\beta}) \leftarrow \{i \in [n] : \boldsymbol{X}_i^\top \boldsymbol{\beta} > 0\}$ be the set of indices that in the linearity part of ReLU function $\max\{0, \boldsymbol{X}_i^\top \boldsymbol{\beta}\}$. The iterative method that used in Section 3.3.2 follows Algorithm 9. The

Algorithm 9 Iterative Heuristic

Input: Set of sample points $\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n \in \mathbb{R}^d \times \mathbb{R}$, initial feasible solution $\boldsymbol{\beta}^0 \in \mathbb{R}^d$, maximum number of iterations T.
Output: A feasible solution $\hat{\boldsymbol{\beta}}$.

1: Initialize $t = 0$.
2: Set the past iterative set set $\mathcal{I}^{-1} \leftarrow \emptyset$.
3: Set the initial iterative set set $\mathcal{I}^0 \leftarrow \mathcal{I}(\beta^0) := \{i \in [n] : \boldsymbol{X}_i^\top \boldsymbol{\beta}^0 > 0\}$
4: Denote the iterative set in t^{th} iteration be $\mathcal{I}^t$.
5: **while** $t < T$ and $\mathcal{I}^t \neq \mathcal{I}^{t-1}$ **do**
6: Set $\boldsymbol{\beta}^{t+1} \leftarrow \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^d} f^\sigma_{\mathcal{I}^t}(\boldsymbol{\beta})$.
7: Set $\mathcal{I}^{t+1} \leftarrow \mathcal{I}(\boldsymbol{\beta}^{t+1})$.
8: Set $t \leftarrow t + 1$.
9: **end while**
10: **return** $\hat{\boldsymbol{\beta}}$ where $\hat{\boldsymbol{\beta}}$ is the final feasible solution obtained in the loop.

iterative heuristic method guarantees the decreasing of objective value in each iteration, i.e., $\min_{\boldsymbol{\beta} \in \mathbb{R}^d} f^\sigma_{\mathcal{I}^t}(\boldsymbol{\beta}) \leq \min_{\boldsymbol{\beta} \in \mathbb{R}^d} f^\sigma_{\mathcal{I}^{t+1}}(\boldsymbol{\beta})$ for $t = 0, 1, 2, \ldots$. Moreover, the algorithm 9 terminates in finite number of iterations.

C.6.3 Appendix: Gradient Descent Method

The gradient descent method that used in Section 3.3.2 and 3.3.2 is Algorithm 10. Note that the outer while-loop follows a standard gradient descent method with gradient $\frac{1}{n} \nabla_{\boldsymbol{\beta}} L(\boldsymbol{\beta}^t)$ and stepsize η_t, and the inner while-loop uses a back search method that guarantee the decreasing of objective value in each outer iteration.

Algorithm 10 Gradient Descent Method

Input: Set of sample points $\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n \in \mathbb{R}^d \times \mathbb{R}$, initial feasible solution $\boldsymbol{\beta}^0 \in \mathbb{R}^d$, maximum number of iterations T, termination criteria parameter $\epsilon > 0$, initial stepsize $\eta_0 > 0$, stepsize parameter $\gamma > 0$, back track parameter $\alpha \in (0, 1)$.
Output: A feasible solution $\hat{\boldsymbol{\beta}}$.

Initialize $t = 0$, $L^{-1} \leftarrow +\infty$, $L(\boldsymbol{\beta}^0) \leftarrow \sum_{i=1}^n (\max\{0, \boldsymbol{X}_i^\top \boldsymbol{\beta}^0\} - Y_i)^2$.
Set $\boldsymbol{\beta}^t$ as the solution obtained in t^{th} iteration.
Set η_t as the stepsize used in t^{th} iteration.
Set $L(\boldsymbol{\beta}) \leftarrow \sum_{i=1}^n (\max\{0, \boldsymbol{X}_i^\top \boldsymbol{\beta}\} - Y_i)^2$.
while $t < T$ and $L(\boldsymbol{\beta}^{t-1}) - L(\boldsymbol{\beta}^t) > \epsilon$ **do**
    Set temporary solution $\bar{\boldsymbol{\beta}}$ be $\bar{\boldsymbol{\beta}} \leftarrow \boldsymbol{\beta}^t - \eta_t \cdot \frac{1}{n} \nabla_{\boldsymbol{\beta}} L(\boldsymbol{\beta}^t)$.
    while $L(\bar{\boldsymbol{\beta}}) \geq L(\boldsymbol{\beta}^t)$ **do**
        Update $\eta_t \leftarrow \alpha \cdot \eta_t$
        Update $\bar{\boldsymbol{\beta}} \leftarrow \boldsymbol{\beta}^t - \eta_t \cdot \frac{1}{n} \nabla_{\boldsymbol{\beta}} L(\boldsymbol{\beta}^t)$.
    end while
    Set $\boldsymbol{\beta}^{t+1} \leftarrow \bar{\boldsymbol{\beta}}$.
    Set $\eta_t \leftarrow \frac{\eta_0}{1+\gamma t}$.
    Set $t \leftarrow t + 1$.
end while
return $\hat{\boldsymbol{\beta}}$ where $\hat{\boldsymbol{\beta}}$ is the final feasible solution obtained in the loop.

C.6.4 Appendix: Stochastic Gradient Descent Method

The stochastic gradient descent method used in this paper is presented below. This algorithm follows a similar updating rule of the gradient descent method (Algorithm 10), the only difference is that in each iteration, the stochastic gradient descent method uniformly picks a mini-batch of size m from the given set of samples $\{\boldsymbol{X}_i\}_{i=1}^n$.

C.7 Main Computational Results, Continued

Figure [C.2, C.3] are the continued numerical results that presented in section 3.3.4 which compare the performance metrics (prediction error, recovery error, generalization error) for various noise levels. Detailed realizable cases in Appendix C.8 provide an empirical result of the performances of the methods for comparison. Below we present the notations that used in each table:

- The first column of each Table in Appendix C.8 is a tuple of 4 elements $(d, n, \rho;$ index$)$

Algorithm 11 Stochastic Gradient Descent Method

Input: Set of sample points $\{(\boldsymbol{X}_i, Y_i)\}_{i=1}^n \in \mathbb{R}^d \times \mathbb{R}$, initial feasible solution $\boldsymbol{\beta}^0 \in \mathbb{R}^d$, maximum number of iterations T, termination criteria parameter $\epsilon > 0$, initial stepsize $\eta_0 > 0$, stepsize parameter $\gamma > 0$, back track parameter $\alpha \in (0, 1)$, size of mini-batch $1 \leq m \leq n$.

Output: A feasible solution $\hat{\boldsymbol{\beta}}$.

Initialize $t = 0$, S^0 uniformly picked from $\{1, \ldots, n\}$ with size m, $L^{-1} \leftarrow +\infty$, $L^0 \leftarrow \sum_{i \in S^0} (\max\{0, \boldsymbol{X}_i^\top \boldsymbol{\beta}^0\} - Y_i)^2$.
Set $\boldsymbol{\beta}^t$ as the solution obtained in t^{th} iteration.
Set η_t as the stepsize used in t^{th} iteration.
Set S^t as the mini-batch of size m in t^{th} iteration.
Set $L(S, \boldsymbol{\beta}) \leftarrow \sum_{i \in S} (\max\{0, \boldsymbol{X}_i^\top \boldsymbol{\beta}\} - Y_i)^2$.
while $t < T$ and $L(S^{t-1}, \boldsymbol{\beta}^{t-1}) - L(S^t, \boldsymbol{\beta}^t) > \epsilon$ **do**
    Set S^{t+1} uniformly from $\{1, \ldots, n\}$ with size m.
    Set temporary solution $\bar{\boldsymbol{\beta}}$ be $\bar{\boldsymbol{\beta}} \leftarrow \boldsymbol{\beta}^t - \eta_t \cdot \frac{1}{m} \nabla_{\boldsymbol{\beta}} L(S^{t+1}, \boldsymbol{\beta}^t)$.
    while $L(\bar{\boldsymbol{\beta}}) \geq L(\boldsymbol{\beta}^t)$ **do**
        Update $\eta_t \leftarrow \alpha \cdot \eta_t$
        Update $\bar{\boldsymbol{\beta}} \leftarrow \boldsymbol{\beta}^t - \eta_t \cdot \frac{1}{m} \nabla_{\boldsymbol{\beta}} L(S^{t+1}, \boldsymbol{\beta}^t)$.
    end while
    Set $\boldsymbol{\beta}^{t+1} \leftarrow \bar{\boldsymbol{\beta}}$.
    Set $\eta_t \leftarrow \frac{\eta_0}{1+\gamma t}$.
    Set $t \leftarrow t + 1$.
end while
return $\hat{\boldsymbol{\beta}}$ where $\hat{\boldsymbol{\beta}}$ is the final feasible solution obtained in the loop.

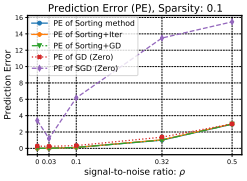
which represents the dimension of $\boldsymbol{\beta}$, the number of training samples, the ratio used for noise ϵ_i, and the index of the instance with such settings respectively.

C.8 Realizable Cases

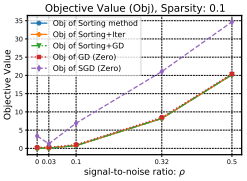
Note that in realizable cases, since the observation samples $\{X_i\}_{i=1}^n$ are constructed to guarantee the full column rank, i.e., the globally optimal solution is unique, then finding a solution with 0 prediction error is equivalent to achieving 0 recovery error. In Figure [C.3c, 3.4c, 3.4g], the averages of the recovery errors of realizable cases are not zero, however their corresponding prediction errors are very small, this may happen when the methods cannot find out the globally optimal solutions. The details of realizable cases are presented in Table [C.1, C.2, C.3, C.4, C.5, C.6, C.7, C.8, C.9, C.10, C.11, C.12, C.13, C.14, C.15].

Table C.1: Realizable Case $d = 10, n = 200$, sparsity $= 0.1$ with $\boldsymbol{\beta}^* \sim N(0.5\mathbf{1}_d, 10\boldsymbol{I}_d)$

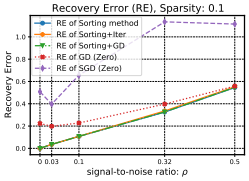
Settings	**Sorting**		**Sorting + Iter**		**Sorting + GD**		**GD**		**SGD**	
	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery
(10, 200, 0.0; 1)	0.0	0.0	0.0	0.0	0.0	0.0	0.373	0.265	1.803	0.601
(10, 200, 0.0; 1)	0.0	0.0	0.0	0.0	0.0	0.0	0.181	0.163	1.433	0.408
(10, 200, 0.0; 2)	0.0	0.0	0.0	0.0	0.0	0.0	0.107	0.108	0.241	0.155
(10, 200, 0.0; 3)	0.0	0.0	0.0	0.0	0.0	0.0	0.32	0.241	1.805	0.541
(10, 200, 0.0; 4)	0.0	0.0	0.0	0.0	0.0	0.0	0.21	0.181	5.446	0.915
(10, 200, 0.0; 5)	0.0	0.0	0.0	0.0	0.0	0.0	0.176	0.179	0.356	0.248
(10, 200, 0.0; 6)	0.0	0.0	0.0	0.0	0.0	0.0	0.154	0.16	5.439	0.754
(10, 200, 0.0; 7)	0.0	0.0	0.0	0.0	0.0	0.0	0.131	0.118	0.257	0.17
(10, 200, 0.0; 8)	0.0	0.0	0.0	0.0	0.0	0.0	0.205	0.173	1.0	0.377
(10, 200, 0.0; 9)	0.0	0.0	0.0	0.0	0.0	0.0	0.137	0.128	0.216	0.16
(10, 200, 0.0; 10)	0.0	0.0	0.0	0.0	0.0	0.0	0.73	0.535	43.452	2.669
(10, 200, 0.0; 11)	0.0	0.0	0.0	0.0	0.0	0.0	0.25	0.227	0.641	0.342
(10, 200, 0.0; 12)	0.0	0.0	0.0	0.0	0.0	0.0	0.207	0.177	0.925	0.36
(10, 200, 0.0; 13)	0.0	0.0	0.0	0.0	0.0	0.0	0.193	0.167	0.706	0.305
(10, 200, 0.0; 14)	0.0	0.0	0.0	0.0	0.0	0.0	0.251	0.227	0.492	0.278
(10, 200, 0.0; 15)	0.0	0.0	0.0	0.0	0.0	0.0	0.669	0.486	1.787	0.65
(10, 200, 0.0; 16)	0.0	0.0	0.0	0.0	0.0	0.0	0.385	0.302	1.607	0.615
(10, 200, 0.0; 17)	0.0	0.0	0.0	0.0	0.0	0.0	0.325	0.244	0.44	0.279
(10, 200, 0.0; 18)	0.0	0.0	0.0	0.0	0.0	0.0	0.494	0.32	1.141	0.503
(10, 200, 0.0; 19)	0.0	0.0	0.0	0.0	0.0	0.0	0.182	0.165	0.323	0.199
(10, 200, 0.0; 20)	0.0	0.0	0.0	0.0	0.0	0.0	0.186	0.16	0.329	0.228



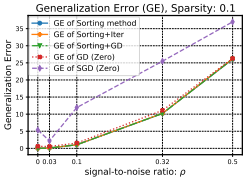
(a) Prediction error



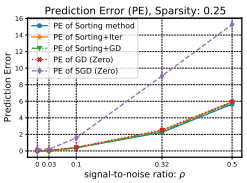
(b) Objective value



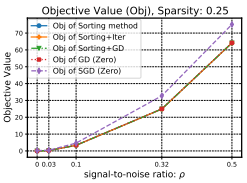
(c) Recovery error



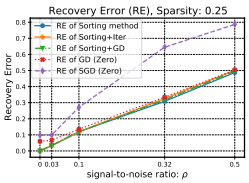
(d) Generalization error



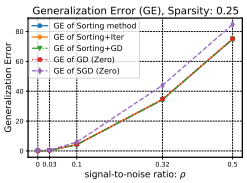
(e) Prediction error



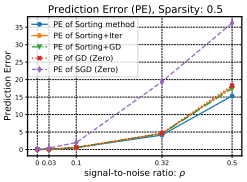
(f) Objective value



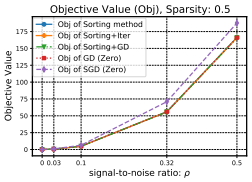
(g) Recovery error



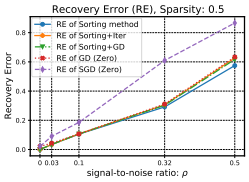
(h) Generalization error



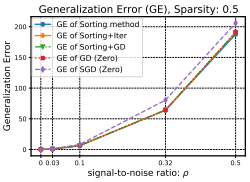
(i) Prediction error



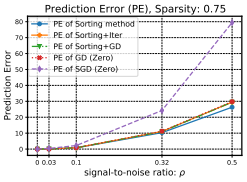
(j) Objective value



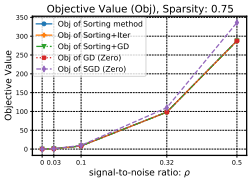
(k) Recovery error



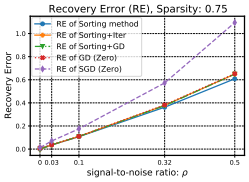
(l) Generalization error



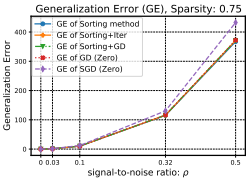
(m) Prediction error



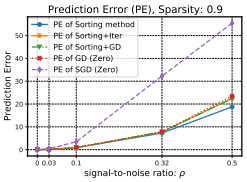
(n) Objective value



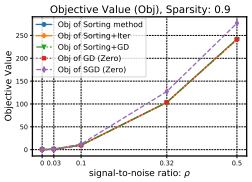
(o) Recovery error



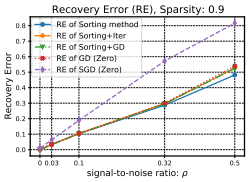
(p) Generalization error



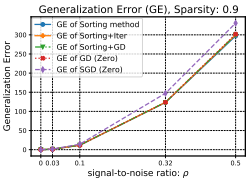
(q) Prediction error



(r) Objective value

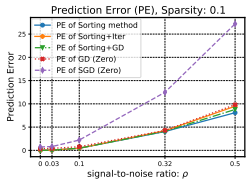


(s) Recovery error

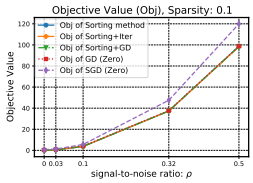


(t) Generalization error

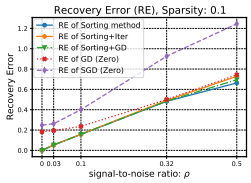
Figure C.2: Numerical Results of sample size $(d, n) = (10, 200)$ and $\boldsymbol{\beta}^* \sim N(0.5 \cdot \mathbf{1}_d, 10 \cdot \boldsymbol{I}_d)$ with sparsity $\{0.1, 0.25, 0.5, 0.75, 0.9\}$



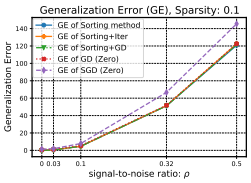
(a) Prediction error



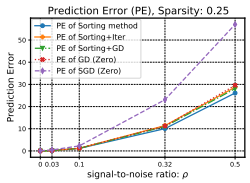
(b) Objective value



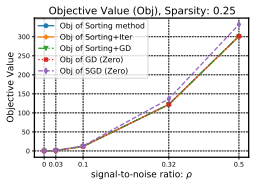
(c) Recovery error



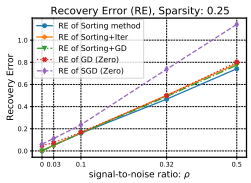
(d) Generalization error



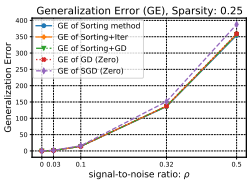
(e) Prediction error



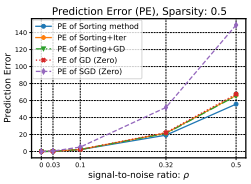
(f) Objective value



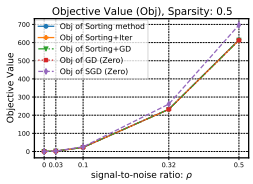
(g) Recovery error



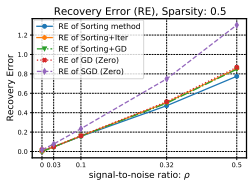
(h) Generalization error



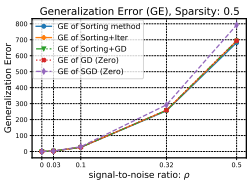
(i) Prediction error



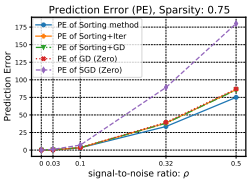
(j) Objective value



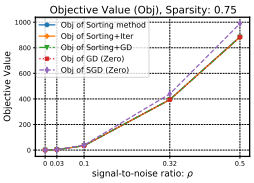
(k) Recovery error



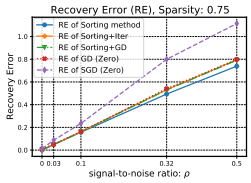
(l) Generalization error



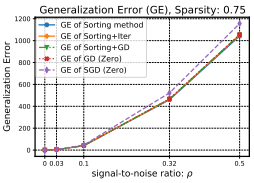
(m) Prediction error



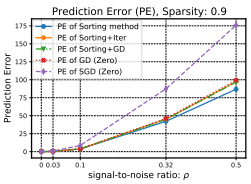
(n) Objective value



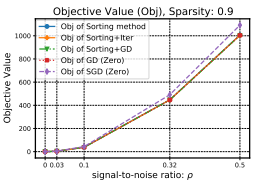
(o) Recovery error



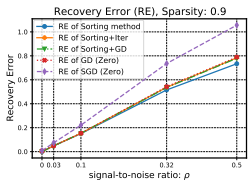
(p) Generalization error



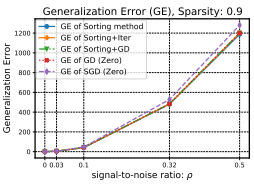
(q) Prediction error



(r) Objective value



(s) Recovery error



(t) Generalization error

Figure C.3: Numerical Results of sample size $(d, n) = (20, 400)$ and $\boldsymbol{\beta}^* \sim N(0.5 \cdot \mathbf{1}_d, 20 \cdot \boldsymbol{I}_d)$ with sparsity $\{0.1, 0.25, 0.5, 0.75, 0.9\}$

Table C.2: Realizable Case $d = 10, n = 200$, sparsity $= 0.25$ with $\boldsymbol{\beta}^* \sim N(0.5\mathbf{1}_d, 10\boldsymbol{I}_d)$

Settings	Sorting		Sorting + Iter		Sorting + GD		GD		SGD	
	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery
(10, 200, 0.0; 1)	0.0	0.0	0.0	0.0	0.0	0.0	0.065	0.075	0.935	0.254
(10, 200, 0.0; 2)	0.0	0.0	0.0	0.0	0.0	0.0	0.046	0.053	0.066	0.064
(10, 200, 0.0; 3)	0.0	0.0	0.0	0.0	0.0	0.0	0.058	0.065	0.185	0.108
(10, 200, 0.0; 4)	0.0	0.0	0.0	0.0	0.0	0.0	0.051	0.058	0.096	0.077
(10, 200, 0.0; 5)	0.0	0.0	0.0	0.0	0.0	0.0	0.05	0.06	0.28	0.14
(10, 200, 0.0; 6)	0.0	0.0	0.0	0.0	0.0	0.0	0.045	0.052	0.038	0.046
(10, 200, 0.0; 7)	0.0	0.0	0.0	0.0	0.0	0.0	0.082	0.086	0.106	0.097
(10, 200, 0.0; 8)	0.0	0.0	0.0	0.0	0.0	0.0	0.038	0.047	0.038	0.047
(10, 200, 0.0; 9)	0.0	0.0	0.0	0.0	0.0	0.0	0.034	0.044	0.042	0.047
(10, 200, 0.0; 10)	0.0	0.0	0.0	0.0	0.0	0.0	0.127	0.114	0.313	0.172
(10, 200, 0.0; 11)	0.0	0.0	0.0	0.0	0.0	0.0	0.052	0.06	0.057	0.06
(10, 200, 0.0; 12)	0.0	0.0	0.0	0.0	0.0	0.0	0.039	0.048	0.027	0.035
(10, 200, 0.0; 13)	0.0	0.0	0.0	0.0	0.0	0.0	0.053	0.063	0.089	0.081
(10, 200, 0.0; 14)	0.0	0.0	0.0	0.0	0.0	0.0	0.058	0.063	1.51	0.31
(10, 200, 0.0; 15)	0.0	0.0	0.0	0.0	0.0	0.0	0.059	0.066	0.135	0.092
(10, 200, 0.0; 16)	0.0	0.0	0.0	0.0	0.0	0.0	0.044	0.052	0.109	0.082
(10, 200, 0.0; 17)	0.0	0.0	0.0	0.0	0.0	0.0	0.039	0.049	0.055	0.056
(10, 200, 0.0; 18)	0.0	0.0	0.0	0.0	0.0	0.0	0.036	0.047	0.049	0.057
(10, 200, 0.0; 19)	0.0	0.0	0.0	0.0	0.0	0.0	0.03	0.04	0.094	0.069
(10, 200, 0.0; 20)	0.0	0.0	0.0	0.0	0.0	0.0	0.041	0.051	0.059	0.06

Table C.3: Realizable Case $d = 10, n = 200$, sparsity $= 0.5$ with $\boldsymbol{\beta}^* \sim N(0.5\mathbf{1}_d, 10\boldsymbol{I}_d)$

Settings	Sorting		Sorting + Iter		Sorting + GD		GD		SGD	
	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery
(10, 200, 0.0; 1)	0.0	0.0	0.0	0.0	0.0	0.0	0.008	0.016	0.051	0.035
(10, 200, 0.0; 2)	0.0	0.0	0.0	0.0	0.0	0.0	0.01	0.018	0.013	0.021
(10, 200, 0.0; 3)	0.0	0.0	0.0	0.0	0.0	0.0	0.02	0.028	0.015	0.022
(10, 200, 0.0; 4)	0.0	0.0	0.0	0.0	0.0	0.0	0.01	0.018	0.007	0.014
(10, 200, 0.0; 5)	0.0	0.0	0.0	0.0	0.0	0.0	0.019	0.027	0.022	0.029
(10, 200, 0.0; 6)	0.0	0.0	0.0	0.0	0.0	0.0	0.009	0.017	0.005	0.012
(10, 200, 0.0; 7)	0.0	0.0	0.0	0.0	0.0	0.0	0.014	0.023	0.036	0.033
(10, 200, 0.0; 8)	0.0	0.0	0.0	0.0	0.0	0.0	0.012	0.02	0.035	0.03
(10, 200, 0.0; 9)	0.0	0.0	0.0	0.0	0.0	0.0	0.009	0.015	0.008	0.013
(10, 200, 0.0; 10)	0.0	0.0	0.0	0.0	0.0	0.0	0.016	0.024	0.009	0.016
(10, 200, 0.0; 11)	0.0	0.0	0.0	0.0	0.0	0.0	0.012	0.02	0.013	0.017
(10, 200, 0.0; 12)	0.0	0.0	0.0	0.0	0.0	0.0	0.009	0.016	0.008	0.012
(10, 200, 0.0; 13)	0.0	0.0	0.0	0.0	0.0	0.0	0.015	0.023	0.012	0.019
(10, 200, 0.0; 14)	0.0	0.0	0.0	0.0	0.0	0.0	0.011	0.018	0.012	0.02
(10, 200, 0.0; 15)	0.0	0.0	0.0	0.0	0.0	0.0	0.019	0.027	0.019	0.023
(10, 200, 0.0; 16)	0.0	0.0	0.0	0.0	0.0	0.0	0.017	0.026	0.006	0.014
(10, 200, 0.0; 17)	0.0	0.0	0.0	0.0	0.0	0.0	0.006	0.013	0.008	0.012
(10, 200, 0.0; 18)	0.0	0.0	0.0	0.0	0.0	0.0	0.021	0.03	0.016	0.025
(10, 200, 0.0; 19)	0.0	0.0	0.0	0.0	0.0	0.0	0.011	0.019	0.034	0.034
(10, 200, 0.0; 20)	0.0	0.0	0.0	0.0	0.0	0.0	0.014	0.023	0.013	0.02

Table C.4: Realizable Case $d = 10, n = 200$, sparsity $= 0.75$ with $\boldsymbol{\beta}^* \sim N(0.5\mathbf{1}_d, 10\boldsymbol{I}_d)$

Settings	Sorting		Sorting + Iter		Sorting + GD		GD		SGD	
	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery
(10, 200, 0.0; 1)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.011	0.007	0.012
(10, 200, 0.0; 2)	0.0	0.0	0.0	0.0	0.0	0.0	0.003	0.008	0.007	0.01
(10, 200, 0.0; 3)	0.0	0.0	0.0	0.0	0.0	0.0	0.002	0.007	0.024	0.02
(10, 200, 0.0; 4)	0.0	0.0	0.0	0.0	0.0	0.0	0.004	0.009	0.002	0.005
(10, 200, 0.0; 5)	0.0	0.0	0.0	0.0	0.0	0.0	0.006	0.013	0.011	0.015
(10, 200, 0.0; 6)	0.0	0.0	0.0	0.0	0.0	0.0	0.004	0.009	0.003	0.007
(10, 200, 0.0; 7)	0.0	0.0	0.0	0.0	0.0	0.0	0.003	0.008	0.033	0.022
(10, 200, 0.0; 8)	0.0	0.0	0.0	0.0	0.0	0.0	0.004	0.01	0.006	0.01
(10, 200, 0.0; 9)	0.0	0.0	0.0	0.0	0.0	0.0	0.002	0.007	0.01	0.012
(10, 200, 0.0; 10)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.011	0.002	0.006
(10, 200, 0.0; 11)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.012	0.009	0.014
(10, 200, 0.0; 12)	0.0	0.0	0.0	0.0	0.0	0.0	0.006	0.011	0.022	0.02
(10, 200, 0.0; 13)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.011	0.004	0.008
(10, 200, 0.0; 14)	0.0	0.0	0.0	0.0	0.0	0.0	0.003	0.009	0.008	0.014
(10, 200, 0.0; 15)	0.0	0.0	0.0	0.0	0.0	0.0	0.006	0.012	0.01	0.013
(10, 200, 0.0; 16)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.011	0.004	0.009
(10, 200, 0.0; 17)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.01	0.013	0.013
(10, 200, 0.0; 18)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.011	0.013	0.013
(10, 200, 0.0; 19)	0.0	0.0	0.0	0.0	0.0	0.0	0.004	0.009	0.003	0.006
(10, 200, 0.0; 20)	0.0	0.0	0.0	0.0	0.0	0.0	0.009	0.015	0.008	0.012

Table C.5: Realizable Case $d = 10, n = 200$, sparsity $= 0.9$ with $\boldsymbol{\beta}^* \sim N(0.5\mathbf{1}_d, 10\boldsymbol{I}_d)$

Settings	Sorting		Sorting + Iter		Sorting + GD		GD		SGD	
	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery
(10, 200, 0.0; 1)	0.0	0.0	0.0	0.0	0.0	0.0	0.002	0.007	0.009	0.011
(10, 200, 0.0; 2)	0.0	0.0	0.0	0.0	0.0	0.0	0.002	0.007	0.007	0.009
(10, 200, 0.0; 3)	0.0	0.0	0.0	0.0	0.0	0.0	0.001	0.004	0.011	0.011
(10, 200, 0.0; 4)	0.0	0.0	0.0	0.0	0.0	0.0	0.001	0.004	0.019	0.016
(10, 200, 0.0; 5)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.011	0.059	0.024
(10, 200, 0.0; 6)	0.0	0.0	0.0	0.0	0.0	0.0	0.001	0.004	0.005	0.008
(10, 200, 0.0; 7)	0.0	0.0	0.0	0.0	0.0	0.0	0.001	0.004	0.006	0.009
(10, 200, 0.0; 8)	0.0	0.0	0.0	0.0	0.0	0.0	0.002	0.007	0.002	0.005
(10, 200, 0.0; 9)	0.0	0.0	0.0	0.0	0.0	0.0	0.003	0.008	0.009	0.011
(10, 200, 0.0; 10)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.003	0.005	0.008
(10, 200, 0.0; 11)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.011	0.013	0.012
(10, 200, 0.0; 12)	0.0	0.0	0.0	0.0	0.0	0.0	0.002	0.005	0.004	0.006
(10, 200, 0.0; 13)	0.0	0.0	0.0	0.0	0.0	0.0	0.002	0.006	0.007	0.009
(10, 200, 0.0; 14)	0.0	0.0	0.0	0.0	0.0	0.0	0.002	0.006	0.004	0.007
(10, 200, 0.0; 15)	0.0	0.0	0.0	0.0	0.0	0.0	0.001	0.005	0.004	0.007
(10, 200, 0.0; 16)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.001	0.001	0.003
(10, 200, 0.0; 17)	0.0	0.0	0.0	0.0	0.0	0.0	0.004	0.009	0.001	0.004
(10, 200, 0.0; 18)	0.0	0.0	0.0	0.0	0.0	0.0	0.002	0.007	0.004	0.006
(10, 200, 0.0; 19)	0.0	0.0	0.0	0.0	0.0	0.0	0.007	0.013	0.008	0.01
(10, 200, 0.0; 20)	0.0	0.0	0.0	0.0	0.0	0.0	0.003	0.008	0.009	0.01

Table C.6: Realizable Case $d = 20, n = 400$, sparsity $= 0.1$ with $\boldsymbol{\beta}^* \sim N(0.5\mathbf{1}_d, 10\boldsymbol{I}_d)$

Settings	Sorting		Sorting + Iter		Sorting + GD		GD		SGD	
	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery
(20, 400, 0.0; 1)	0.0	0.0	0.0	0.0	0.0	0.0	0.277	0.147	0.343	0.153
(20, 400, 0.0; 2)	0.0	0.0	0.0	0.0	0.0	0.0	0.336	0.179	0.715	0.258
(20, 400, 0.0; 3)	0.0	0.0	0.0	0.0	0.0	0.0	0.212	0.121	0.503	0.187
(20, 400, 0.0; 4)	0.0	0.0	0.0	0.0	0.0	0.0	0.358	0.169	0.558	0.206
(20, 400, 0.0; 5)	0.0	0.0	0.0	0.0	0.0	0.0	0.263	0.141	0.616	0.21
(20, 400, 0.0; 6)	0.0	0.0	0.0	0.0	0.0	0.0	0.392	0.189	1.385	0.348
(20, 400, 0.0; 7)	0.0	0.0	0.0	0.0	0.0	0.0	0.269	0.145	0.369	0.164
(20, 400, 0.0; 8)	0.0	0.0	0.0	0.0	0.0	0.0	0.339	0.171	0.45	0.196
(20, 400, 0.0; 9)	0.0	0.0	0.0	0.0	0.0	0.0	0.515	0.237	0.86	0.306
(20, 400, 0.0; 10)	0.0	0.0	0.0	0.0	0.0	0.0	0.354	0.177	0.568	0.222
(20, 400, 0.0; 11)	0.0	0.0	0.0	0.0	0.0	0.0	0.518	0.238	0.843	0.293
(20, 400, 0.0; 12)	0.0	0.0	0.0	0.0	0.0	0.0	0.334	0.176	1.108	0.315
(20, 400, 0.0; 13)	0.0	0.0	0.0	0.0	0.0	0.0	0.368	0.184	0.611	0.232
(20, 400, 0.0; 14)	0.0	0.0	0.0	0.0	0.0	0.0	0.432	0.217	1.237	0.346
(20, 400, 0.0; 15)	0.0	0.0	0.0	0.0	0.0	0.0	0.461	0.211	0.99	0.304
(20, 400, 0.0; 16)	0.0	0.0	0.0	0.0	0.0	0.0	0.338	0.171	0.357	0.168
(20, 400, 0.0; 17)	0.0	0.0	0.0	0.0	0.0	0.0	0.445	0.212	0.615	0.244
(20, 400, 0.0; 18)	0.0	0.0	0.0	0.0	0.0	0.0	0.469	0.224	0.776	0.29
(20, 400, 0.0; 19)	0.0	0.0	0.0	0.0	0.0	0.0	0.275	0.151	0.35	0.161
(20, 400, 0.0; 20)	0.0	0.0	0.0	0.0	0.0	0.0	0.417	0.205	1.337	0.349

Table C.7: Realizable Case $d = 20, n = 400$, sparsity $= 0.25$ with $\boldsymbol{\beta}^* \sim N(0.5\mathbf{1}_d, 10\boldsymbol{I}_d)$

Settings	Sorting		Sorting + Iter		Sorting + GD		GD		SGD	
	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery
(20, 400, 0.0; 1)	0.0	0.0	0.0	0.0	0.0	0.0	0.044	0.035	0.06	0.04
(20, 400, 0.0; 2)	0.0	0.0	0.0	0.0	0.0	0.0	0.035	0.031	0.083	0.047
(20, 400, 0.0; 3)	0.0	0.0	0.0	0.0	0.0	0.0	0.086	0.063	0.121	0.072
(20, 400, 0.0; 4)	0.0	0.0	0.0	0.0	0.0	0.0	0.08	0.057	0.093	0.06
(20, 400, 0.0; 5)	0.0	0.0	0.0	0.0	0.0	0.0	0.054	0.043	0.089	0.052
(20, 400, 0.0; 6)	0.0	0.0	0.0	0.0	0.0	0.0	0.039	0.034	0.042	0.037
(20, 400, 0.0; 7)	0.0	0.0	0.0	0.0	0.0	0.0	0.057	0.044	0.133	0.064
(20, 400, 0.0; 8)	0.0	0.0	0.0	0.0	0.0	0.0	0.073	0.054	0.067	0.049
(20, 400, 0.0; 9)	0.0	0.0	0.0	0.0	0.0	0.0	0.094	0.062	0.185	0.087
(20, 400, 0.0; 10)	0.0	0.0	0.0	0.0	0.0	0.0	0.078	0.058	0.128	0.071
(20, 400, 0.0; 11)	0.0	0.0	0.0	0.0	0.0	0.0	0.07	0.052	0.198	0.085
(20, 400, 0.0; 12)	0.0	0.0	0.0	0.0	0.0	0.0	0.06	0.046	0.097	0.059
(20, 400, 0.0; 13)	0.0	0.0	0.0	0.0	0.0	0.0	0.066	0.048	0.079	0.051
(20, 400, 0.0; 14)	0.0	0.0	0.0	0.0	0.0	0.0	0.065	0.049	0.075	0.052
(20, 400, 0.0; 15)	0.0	0.0	0.0	0.0	0.0	0.0	0.064	0.048	0.069	0.048
(20, 400, 0.0; 16)	0.0	0.0	0.0	0.0	0.0	0.0	0.08	0.057	0.106	0.063
(20, 400, 0.0; 17)	0.0	0.0	0.0	0.0	0.0	0.0	0.082	0.057	0.119	0.067
(20, 400, 0.0; 18)	0.0	0.0	0.0	0.0	0.0	0.0	0.068	0.051	0.152	0.076
(20, 400, 0.0; 19)	0.0	0.0	0.0	0.0	0.0	0.0	0.073	0.055	0.158	0.082
(20, 400, 0.0; 20)	0.0	0.0	0.0	0.0	0.0	0.0	0.048	0.039	0.092	0.052

Table C.8: Realizable Case $d = 20, n = 400$, sparsity $= 0.5$ with $\boldsymbol{\beta}^* \sim N(0.5\mathbf{1}_d, 10\boldsymbol{I}_d)$

Settings	Sorting		Sorting + Iter		Sorting + GD		GD		SGD	
	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery
(20, 400, 0.0; 1)	0.0	0.0	0.0	0.0	0.0	0.0	0.015	0.017	0.018	0.016
(20, 400, 0.0; 2)	0.0	0.0	0.0	0.0	0.0	0.0	0.015	0.016	0.01	0.011
(20, 400, 0.0; 3)	0.0	0.0	0.0	0.0	0.0	0.0	0.019	0.019	0.015	0.014
(20, 400, 0.0; 4)	0.0	0.0	0.0	0.0	0.0	0.0	0.019	0.019	0.025	0.019
(20, 400, 0.0; 5)	0.0	0.0	0.0	0.0	0.0	0.0	0.023	0.022	0.023	0.019
(20, 400, 0.0; 6)	0.0	0.0	0.0	0.0	0.0	0.0	0.013	0.015	0.011	0.013
(20, 400, 0.0; 7)	0.0	0.0	0.0	0.0	0.0	0.0	0.026	0.025	0.106	0.044
(20, 400, 0.0; 8)	0.0	0.0	0.0	0.0	0.0	0.0	0.024	0.023	0.027	0.022
(20, 400, 0.0; 9)	0.0	0.0	0.0	0.0	0.0	0.0	0.029	0.027	0.01	0.014
(20, 400, 0.0; 10)	0.0	0.0	0.0	0.0	0.0	0.0	0.017	0.018	0.01	0.012
(20, 400, 0.0; 11)	0.0	0.0	0.0	0.0	0.0	0.0	0.018	0.019	0.026	0.02
(20, 400, 0.0; 12)	0.0	0.0	0.0	0.0	0.0	0.0	0.013	0.015	0.028	0.021
(20, 400, 0.0; 13)	0.0	0.0	0.0	0.0	0.0	0.0	0.016	0.017	0.023	0.018
(20, 400, 0.0; 14)	0.0	0.0	0.0	0.0	0.0	0.0	0.012	0.015	0.013	0.015
(20, 400, 0.0; 15)	0.0	0.0	0.0	0.0	0.0	0.0	0.02	0.02	0.021	0.019
(20, 400, 0.0; 16)	0.0	0.0	0.0	0.0	0.0	0.0	0.013	0.015	0.011	0.012
(20, 400, 0.0; 17)	0.0	0.0	0.0	0.0	0.0	0.0	0.017	0.017	0.01	0.012
(20, 400, 0.0; 18)	0.0	0.0	0.0	0.0	0.0	0.0	0.03	0.027	0.053	0.031
(20, 400, 0.0; 19)	0.0	0.0	0.0	0.0	0.0	0.0	0.011	0.013	0.009	0.01
(20, 400, 0.0; 20)	0.0	0.0	0.0	0.0	0.0	0.0	0.017	0.017	0.01	0.011

Table C.9: Realizable Case $d = 20, n = 400$, sparsity $= 0.75$ with $\boldsymbol{\beta}^* \sim N(0.5\mathbf{1}_d, 10\boldsymbol{I}_d)$

Settings	Sorting		Sorting + Iter		Sorting + GD		GD		SGD	
	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery
(20, 400, 0.0; 1)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.008	0.005	0.006
(20, 400, 0.0; 2)	0.0	0.0	0.0	0.0	0.0	0.0	0.017	0.017	0.03	0.019
(20, 400, 0.0; 3)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.008	0.009	0.008
(20, 400, 0.0; 4)	0.0	0.0	0.0	0.0	0.0	0.0	0.011	0.014	0.013	0.011
(20, 400, 0.0; 5)	0.0	0.0	0.0	0.0	0.0	0.0	0.007	0.01	0.016	0.013
(20, 400, 0.0; 6)	0.0	0.0	0.0	0.0	0.0	0.0	0.008	0.01	0.009	0.008
(20, 400, 0.0; 7)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.007	0.013	0.01
(20, 400, 0.0; 8)	0.0	0.0	0.0	0.0	0.0	0.0	0.003	0.006	0.005	0.006
(20, 400, 0.0; 9)	0.0	0.0	0.0	0.0	0.0	0.0	0.011	0.013	0.022	0.015
(20, 400, 0.0; 10)	0.0	0.0	0.0	0.0	0.0	0.0	0.01	0.012	0.02	0.014
(20, 400, 0.0; 11)	0.0	0.0	0.0	0.0	0.0	0.0	0.012	0.014	0.03	0.019
(20, 400, 0.0; 12)	0.0	0.0	0.0	0.0	0.0	0.0	0.008	0.01	0.013	0.01
(20, 400, 0.0; 13)	0.0	0.0	0.0	0.0	0.0	0.0	0.01	0.012	0.008	0.009
(20, 400, 0.0; 14)	0.0	0.0	0.0	0.0	0.0	0.0	0.012	0.014	0.021	0.015
(20, 400, 0.0; 15)	0.0	0.0	0.0	0.0	0.0	0.0	0.008	0.01	0.025	0.015
(20, 400, 0.0; 16)	0.0	0.0	0.0	0.0	0.0	0.0	0.004	0.007	0.017	0.012
(20, 400, 0.0; 17)	0.0	0.0	0.0	0.0	0.0	0.0	0.006	0.009	0.007	0.008
(20, 400, 0.0; 18)	0.0	0.0	0.0	0.0	0.0	0.0	0.007	0.01	0.013	0.011
(20, 400, 0.0; 19)	0.0	0.0	0.0	0.0	0.0	0.0	0.008	0.01	0.01	0.01
(20, 400, 0.0; 20)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.008	0.007	0.007

Table C.10: Realizable Case $d = 20, n = 400$, sparsity $= 0.9$ with $\boldsymbol{\beta}^* \sim N(0.5\mathbf{1}_d, 10\boldsymbol{I}_d)$

Settings	**Sorting**		**Sorting + Iter**		**Sorting + GD**		**GD**		**SGD**	
	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery
(20, 400, 0.0; 1)	0.0	0.0	0.0	0.0	0.0	0.0	0.003	0.005	0.005	0.005
(20, 400, 0.0; 2)	0.0	0.0	0.0	0.0	0.0	0.0	0.003	0.005	0.018	0.011
(20, 400, 0.0; 3)	0.0	0.0	0.0	0.0	0.0	0.0	0.003	0.006	0.005	0.006
(20, 400, 0.0; 4)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.007	0.01	0.007
(20, 400, 0.0; 5)	0.0	0.0	0.0	0.0	0.0	0.0	0.006	0.009	0.022	0.014
(20, 400, 0.0; 6)	0.0	0.0	0.0	0.0	0.0	0.0	0.006	0.009	0.004	0.005
(20, 400, 0.0; 7)	0.0	0.0	0.0	0.0	0.0	0.0	0.004	0.007	0.007	0.008
(20, 400, 0.0; 8)	0.0	0.0	0.0	0.0	0.0	0.0	0.003	0.006	0.004	0.005
(20, 400, 0.0; 9)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.007	0.015	0.01
(20, 400, 0.0; 10)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.007	0.005	0.005
(20, 400, 0.0; 11)	0.0	0.0	0.0	0.0	0.0	0.0	0.004	0.007	0.006	0.006
(20, 400, 0.0; 12)	0.0	0.0	0.0	0.0	0.0	0.0	0.006	0.008	0.013	0.009
(20, 400, 0.0; 13)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.007	0.015	0.01
(20, 400, 0.0; 14)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.007	0.005	0.006
(20, 400, 0.0; 15)	0.0	0.0	0.0	0.0	0.0	0.0	0.003	0.006	0.006	0.005
(20, 400, 0.0; 16)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.002	0.004	0.005
(20, 400, 0.0; 17)	0.0	0.0	0.0	0.0	0.0	0.0	0.002	0.005	0.01	0.008
(20, 400, 0.0; 18)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.007	0.008	0.007
(20, 400, 0.0; 19)	0.0	0.0	0.0	0.0	0.0	0.0	0.006	0.008	0.006	0.006
(20, 400, 0.0; 20)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.009	0.011	0.009

Table C.11: Realizable Case $d = 50, n = 1000$, sparsity $= 0.1$ with $\boldsymbol{\beta}^* \sim N(0.5\mathbf{1}_d, 10\boldsymbol{I}_d)$

Settings	**Sorting**		**Sorting + Iter**		**Sorting + GD**		**GD**		**SGD**	
	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery
(50, 1000, 0.0; 1)	0.0	0.0	0.0	0.0	0.0	0.0	0.417	0.118	0.571	0.137
(50, 1000, 0.0; 2)	0.0	0.0	0.0	0.0	0.0	0.0	0.438	0.12	0.479	0.127
(50, 1000, 0.0; 3)	0.0	0.0	0.0	0.0	0.0	0.0	0.479	0.128	0.689	0.15
(50, 1000, 0.0; 4)	0.0	0.0	0.0	0.0	0.0	0.0	0.472	0.126	0.705	0.153
(50, 1000, 0.0; 5)	0.0	0.0	0.0	0.0	0.0	0.0	0.585	0.153	0.707	0.165
(50, 1000, 0.0; 6)	0.0	0.0	0.0	0.0	0.0	0.0	0.349	0.105	0.474	0.122
(50, 1000, 0.0; 7)	0.0	0.0	0.0	0.0	0.0	0.0	0.647	0.162	0.875	0.188
(50, 1000, 0.0; 8)	0.0	0.0	0.0	0.0	0.0	0.0	0.516	0.14	0.534	0.138
(50, 1000, 0.0; 9)	0.0	0.0	0.0	0.0	0.0	0.0	0.502	0.137	0.673	0.158
(50, 1000, 0.0; 10)	0.0	0.0	0.0	0.0	0.0	0.0	0.525	0.136	0.591	0.142
(50, 1000, 0.0; 11)	0.0	0.0	0.0	0.0	0.0	0.0	0.596	0.147	0.84	0.174
(50, 1000, 0.0; 12)	0.0	0.0	0.0	0.0	0.0	0.0	0.467	0.126	0.533	0.132
(50, 1000, 0.0; 13)	0.0	0.0	0.0	0.0	0.0	0.0	0.732	0.169	0.94	0.187
(50, 1000, 0.0; 14)	0.0	0.0	0.0	0.0	0.0	0.0	0.417	0.116	0.533	0.129
(50, 1000, 0.0; 15)	0.0	0.0	0.0	0.0	0.0	0.0	0.527	0.142	0.609	0.15
(50, 1000, 0.0; 16)	0.0	0.0	0.0	0.0	0.0	0.0	0.465	0.13	0.538	0.141
(50, 1000, 0.0; 17)	0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.148	0.895	0.178
(50, 1000, 0.0; 18)	0.0	0.0	0.0	0.0	0.0	0.0	0.518	0.137	0.797	0.172
(50, 1000, 0.0; 19)	0.0	0.0	0.0	0.0	0.0	0.0	0.509	0.134	0.65	0.15
(50, 1000, 0.0; 20)	0.0	0.0	0.0	0.0	0.0	0.0	0.505	0.132	0.656	0.149

Table C.12: Realizable Case $d = 50, n = 1000$, sparsity $= 0.25$ with $\boldsymbol{\beta}^* \sim N(0.5\mathbf{1}_d, 10\boldsymbol{I}_d)$

Settings	Sorting		Sorting + Iter		Sorting + GD		GD		SGD	
	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery
(50, 1000, 0.0; 1)	0.0	0.0	0.0	0.0	0.0	0.0	0.07	0.032	0.091	0.035
(50, 1000, 0.0; 2)	0.0	0.0	0.0	0.0	0.0	0.0	0.088	0.037	0.079	0.033
(50, 1000, 0.0; 3)	0.0	0.0	0.0	0.0	0.0	0.0	0.072	0.032	0.079	0.032
(50, 1000, 0.0; 4)	0.0	0.0	0.0	0.0	0.0	0.0	0.106	0.043	0.145	0.048
(50, 1000, 0.0; 5)	0.0	0.0	0.0	0.0	0.0	0.0	0.069	0.031	0.076	0.032
(50, 1000, 0.0; 6)	0.0	0.0	0.0	0.0	0.0	0.0	0.076	0.033	0.113	0.039
(50, 1000, 0.0; 7)	0.0	0.0	0.0	0.0	0.0	0.0	0.106	0.042	0.115	0.042
(50, 1000, 0.0; 8)	0.0	0.0	0.0	0.0	0.0	0.0	0.083	0.036	0.093	0.036
(50, 1000, 0.0; 9)	0.0	0.0	0.0	0.0	0.0	0.0	0.085	0.035	0.095	0.037
(50, 1000, 0.0; 10)	0.0	0.0	0.0	0.0	0.0	0.0	0.065	0.03	0.067	0.03
(50, 1000, 0.0; 11)	0.0	0.0	0.0	0.0	0.0	0.0	0.074	0.033	0.067	0.028
(50, 1000, 0.0; 12)	0.0	0.0	0.0	0.0	0.0	0.0	0.079	0.034	0.067	0.029
(50, 1000, 0.0; 13)	0.0	0.0	0.0	0.0	0.0	0.0	0.072	0.032	0.102	0.037
(50, 1000, 0.0; 14)	0.0	0.0	0.0	0.0	0.0	0.0	0.114	0.045	0.132	0.046
(50, 1000, 0.0; 15)	0.0	0.0	0.0	0.0	0.0	0.0	0.085	0.036	0.1	0.037
(50, 1000, 0.0; 16)	0.0	0.0	0.0	0.0	0.0	0.0	0.082	0.035	0.133	0.045
(50, 1000, 0.0; 17)	0.0	0.0	0.0	0.0	0.0	0.0	0.084	0.036	0.086	0.035
(50, 1000, 0.0; 18)	0.0	0.0	0.0	0.0	0.0	0.0	0.084	0.036	0.093	0.037
(50, 1000, 0.0; 19)	0.0	0.0	0.0	0.0	0.0	0.0	0.091	0.037	0.09	0.036
(50, 1000, 0.0; 20)	0.0	0.0	0.0	0.0	0.0	0.0	0.079	0.034	0.09	0.034

Table C.13: Realizable Case $d = 50, n = 1000$, sparsity $= 0.5$ with $\boldsymbol{\beta}^* \sim N(0.5\mathbf{1}_d, 10\boldsymbol{I}_d)$

Settings	Sorting		Sorting + Iter		Sorting + GD		GD		SGD	
	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery
(50, 1000, 0.0; 1)	0.0	0.0	0.0	0.0	0.0	0.0	0.032	0.018	0.038	0.017
(50, 1000, 0.0; 2)	0.0	0.0	0.0	0.0	0.0	0.0	0.014	0.01	0.02	0.012
(50, 1000, 0.0; 3)	0.0	0.0	0.0	0.0	0.0	0.0	0.018	0.012	0.057	0.019
(50, 1000, 0.0; 4)	0.0	0.0	0.0	0.0	0.0	0.0	0.023	0.014	0.025	0.014
(50, 1000, 0.0; 5)	0.0	0.0	0.0	0.0	0.0	0.0	0.029	0.016	0.041	0.017
(50, 1000, 0.0; 6)	0.0	0.0	0.0	0.0	0.0	0.0	0.026	0.015	0.026	0.013
(50, 1000, 0.0; 7)	0.0	0.0	0.0	0.0	0.0	0.0	0.023	0.013	0.019	0.011
(50, 1000, 0.0; 8)	0.0	0.0	0.0	0.0	0.0	0.0	0.023	0.013	0.028	0.013
(50, 1000, 0.0; 9)	0.0	0.0	0.0	0.0	0.0	0.0	0.026	0.015	0.036	0.015
(50, 1000, 0.0; 10)	0.0	0.0	0.0	0.0	0.0	0.0	0.019	0.011	0.018	0.01
(50, 1000, 0.0; 11)	0.0	0.0	0.0	0.0	0.0	0.0	0.016	0.011	0.026	0.013
(50, 1000, 0.0; 12)	0.0	0.0	0.0	0.0	0.0	0.0	0.028	0.015	0.017	0.01
(50, 1000, 0.0; 13)	0.0	0.0	0.0	0.0	0.0	0.0	0.022	0.014	0.037	0.016
(50, 1000, 0.0; 14)	0.0	0.0	0.0	0.0	0.0	0.0	0.021	0.013	0.036	0.015
(50, 1000, 0.0; 15)	0.0	0.0	0.0	0.0	0.0	0.0	0.023	0.014	0.037	0.017
(50, 1000, 0.0; 16)	0.0	0.0	0.0	0.0	0.0	0.0	0.025	0.014	0.023	0.013
(50, 1000, 0.0; 17)	0.0	0.0	0.0	0.0	0.0	0.0	0.016	0.011	0.011	0.008
(50, 1000, 0.0; 18)	0.0	0.0	0.0	0.0	0.0	0.0	0.019	0.012	0.041	0.015
(50, 1000, 0.0; 19)	0.0	0.0	0.0	0.0	0.0	0.0	0.028	0.016	0.046	0.019
(50, 1000, 0.0; 20)	0.0	0.0	0.0	0.0	0.0	0.0	0.031	0.017	0.034	0.016

Table C.14: Realizable Case $d = 50, n = 1000$, sparsity $= 0.75$ with $\boldsymbol{\beta}^* \sim N(0.5\mathbf{1}_d, 10\boldsymbol{I}_d)$

Settings	Sorting		Sorting + Iter		Sorting + GD		GD		SGD	
	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery
(50, 1000, 0.0; 1)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.005	0.006	0.004
(50, 1000, 0.0; 2)	0.0	0.0	0.0	0.0	0.0	0.0	0.009	0.007	0.011	0.006
(50, 1000, 0.0; 3)	0.0	0.0	0.0	0.0	0.0	0.0	0.009	0.007	0.01	0.006
(50, 1000, 0.0; 4)	0.0	0.0	0.0	0.0	0.0	0.0	0.009	0.007	0.013	0.007
(50, 1000, 0.0; 5)	0.0	0.0	0.0	0.0	0.0	0.0	0.007	0.006	0.009	0.006
(50, 1000, 0.0; 6)	0.0	0.0	0.0	0.0	0.0	0.0	0.011	0.008	0.008	0.005
(50, 1000, 0.0; 7)	0.0	0.0	0.0	0.0	0.0	0.0	0.008	0.006	0.009	0.006
(50, 1000, 0.0; 8)	0.0	0.0	0.0	0.0	0.0	0.0	0.008	0.007	0.006	0.005
(50, 1000, 0.0; 9)	0.0	0.0	0.0	0.0	0.0	0.0	0.007	0.006	0.008	0.006
(50, 1000, 0.0; 10)	0.0	0.0	0.0	0.0	0.0	0.0	0.007	0.006	0.006	0.005
(50, 1000, 0.0; 11)	0.0	0.0	0.0	0.0	0.0	0.0	0.007	0.006	0.006	0.005
(50, 1000, 0.0; 12)	0.0	0.0	0.0	0.0	0.0	0.0	0.007	0.006	0.007	0.006
(50, 1000, 0.0; 13)	0.0	0.0	0.0	0.0	0.0	0.0	0.01	0.007	0.014	0.007
(50, 1000, 0.0; 14)	0.0	0.0	0.0	0.0	0.0	0.0	0.006	0.005	0.015	0.007
(50, 1000, 0.0; 15)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.005	0.013	0.006
(50, 1000, 0.0; 16)	0.0	0.0	0.0	0.0	0.0	0.0	0.007	0.006	0.008	0.005
(50, 1000, 0.0; 17)	0.0	0.0	0.0	0.0	0.0	0.0	0.006	0.006	0.008	0.006
(50, 1000, 0.0; 18)	0.0	0.0	0.0	0.0	0.0	0.0	0.007	0.006	0.014	0.007
(50, 1000, 0.0; 19)	0.0	0.0	0.0	0.0	0.0	0.0	0.01	0.008	0.025	0.01
(50, 1000, 0.0; 20)	0.0	0.0	0.0	0.0	0.0	0.0	0.01	0.007	0.011	0.006

Table C.15: Realizable Case $d = 50, n = 1000$, sparsity $= 0.9$ with $\boldsymbol{\beta}^* \sim N(0.5\mathbf{1}_d, 10\boldsymbol{I}_d)$

Settings	Sorting		Sorting + Iter		Sorting + GD		GD		SGD	
	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery	Prediction	Recovery
(50, 1000, 0.0; 1)	0.0	0.0	0.0	0.0	0.0	0.0	0.004	0.004	0.01	0.005
(50, 1000, 0.0; 2)	0.0	0.0	0.0	0.0	0.0	0.0	0.006	0.006	0.005	0.004
(50, 1000, 0.0; 3)	0.0	0.0	0.0	0.0	0.0	0.0	0.003	0.004	0.007	0.004
(50, 1000, 0.0; 4)	0.0	0.0	0.0	0.0	0.0	0.0	0.004	0.004	0.005	0.003
(50, 1000, 0.0; 5)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.004	0.008	0.005
(50, 1000, 0.0; 6)	0.0	0.0	0.0	0.0	0.0	0.0	0.007	0.006	0.01	0.005
(50, 1000, 0.0; 7)	0.0	0.0	0.0	0.0	0.0	0.0	0.006	0.005	0.013	0.006
(50, 1000, 0.0; 8)	0.0	0.0	0.0	0.0	0.0	0.0	0.004	0.004	0.011	0.005
(50, 1000, 0.0; 9)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.004	0.01	0.005
(50, 1000, 0.0; 10)	0.0	0.0	0.0	0.0	0.0	0.0	0.003	0.003	0.011	0.006
(50, 1000, 0.0; 11)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.005	0.008	0.005
(50, 1000, 0.0; 12)	0.0	0.0	0.0	0.0	0.0	0.0	0.007	0.006	0.01	0.005
(50, 1000, 0.0; 13)	0.0	0.0	0.0	0.0	0.0	0.0	0.007	0.006	0.009	0.005
(50, 1000, 0.0; 14)	0.0	0.0	0.0	0.0	0.0	0.0	0.002	0.003	0.009	0.005
(50, 1000, 0.0; 15)	0.0	0.0	0.0	0.0	0.0	0.0	0.006	0.005	0.013	0.006
(50, 1000, 0.0; 16)	0.0	0.0	0.0	0.0	0.0	0.0	0.005	0.005	0.01	0.005
(50, 1000, 0.0; 17)	0.0	0.0	0.0	0.0	0.0	0.0	0.006	0.005	0.01	0.005
(50, 1000, 0.0; 18)	0.0	0.0	0.0	0.0	0.0	0.0	0.004	0.004	0.016	0.007
(50, 1000, 0.0; 19)	0.0	0.0	0.0	0.0	0.0	0.0	0.007	0.006	0.007	0.005
(50, 1000, 0.0; 20)	0.0	0.0	0.0	0.0	0.0	0.0	0.004	0.004	0.005	0.004

REFERENCES

[1] S. S. Dey, R. Mazumder, and G. Wang, "A convex integer programming approach for optimal sparse pca," *arXiv preprint arXiv:1810.09062*, 2018.

[2] J. Cadima and I. T. Jolliffe, "Loading and correlations in the interpretation of principle compenents," *Journal of applied Statistics*, vol. 22, no. 2, pp. 203–214, 1995.

[3] I. T. Jolliffe, N. T. Trendafilov, and M. Uddin, "A modified principal component technique based on the LASSO," *Journal of computational and Graphical Statistics*, vol. 12, no. 3, pp. 531–547, 2003.

[4] H. Zou, T. Hastie, and R. Tibshirani, "Sparse principal component analysis," *Journal of computational and graphical statistics*, vol. 15, no. 2, pp. 265–286, 2006.

[5] G. I. Allen and M. Maletić-Savatić, "Sparse non-negative generalized pca with applications to metabolomics," *Bioinformatics*, vol. 27, no. 21, pp. 3029–3035, 2011.

[6] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical learning with sparsity: the lasso and generalizations*. CRC press, 2015.

[7] D. M. Witten, R. Tibshirani, and T. Hastie, "A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis," *Biostatistics*, vol. 10, no. 3, pp. 515–534, 2009.

[8] Z. Zhang, H. Zha, and H. Simon, "Low-rank approximations with sparse factors I: Basic algorithms and error analysis," *SIAM Journal on Matrix Analysis and Applications*, vol. 23, no. 3, pp. 706–727, 2002.

[9] Y. He, R. D. Monteiro, and H. Park, "An algorithm for sparse PCA based on a new sparsity control criterion," in *Proceedings of the 2011 SIAM International Conference on Data Mining*, SIAM, 2011, pp. 771–782.

[10] A. d'Aspremont, F. Bach, and L. E. Ghaoui, "Optimal solutions for sparse principal component analysis," *Journal of Machine Learning Research*, vol. 9, no. Jul, pp. 1269–1294, 2008.

[11] X.-T. Yuan and T. Zhang, "Truncated power method for sparse eigenvalue problems," *Journal of Machine Learning Research*, vol. 14, no. Apr, pp. 899–925, 2013.

[12] M. Journée, Y. Nesterov, P. Richtárik, and R. Sepulchre, "Generalized power method for sparse principal component analysis," *Journal of Machine Learning Research*, vol. 11, no. Feb, pp. 517–553, 2010.

[13] S. O. Chan, D. Papailiopoulos, and A. Rubinstein, "On the worst-case approximability of sparse PCA," *arXiv preprint arXiv:1507.05950*, 2015.

[14] M. Magdon-Ismail, "NP-hardness and inapproximability of sparse PCA," *Information Processing Letters*, vol. 126, pp. 35–38, 2017.

[15] A. Chowdhury, P. Drineas, D. P. Woodruff, and S. Zhou, "Approximation algorithms for sparse principal component analysis," *arXiv preprint arXiv:2006.12748*, 2020.

[16] D. Papailiopoulos, A. Dimakis, and S. Korokythakis, "Sparse PCA through low-rank approximations," in *International Conference on Machine Learning*, 2013, pp. 747–755.

[17] Q. Berthet and P. Rigollet, "Optimal detection of sparse principal components in high dimension," *Annals of Statistics*, vol. 41, no. 4, pp. 1780–1815, 2013.

[18] Z. Wang, C. T. Bauch, S. Bhattacharyya, A. d'Onofrio, P. Manfredi, M. Perc, N. Perra, M. Salathé, and D. Zhao, "Statistical physics of vaccination," *Physics Reports*, vol. 664, pp. 1–113, 2016.

[19] A. d'Aspremont, L. El Ghaoui, M. I. Jordan, and G. R. Lanckriet, "A direct formulation for sparse pca using semidefinite programming," *SIAM review*, vol. 49, no. 3, pp. 434–448, 2007.

[20] R. Mazumder and P. Radchenko, "The discrete dantzig selector: Estimating sparse linear models via mixed integer linear optimization," *IEEE Transactions on Information Theory*, vol. 63, no. 5, pp. 3053–3075, 2017.

[21] A. d'Aspremont, L. E. Ghaoui, M. I. Jordan, and G. R. Lanckriet, "A direct formulation for sparse PCA using semidefinite programming," in *Advances in neural information processing systems*, 2005, pp. 41–48.

[22] A. d'Aspremont, F. Bach, and L. El Ghaoui, "Approximation bounds for sparse principal component analysis," *Mathematical Programming*, vol. 148, no. 1-2, pp. 89–110, 2014.

[23] Y. Zhang, A. d'Aspremont, and L. El Ghaoui, "Sparse PCA: Convex relaxations, algorithms and applications," in *Handbook on Semidefinite, Conic and Polynomial Optimization*, Springer, 2012, pp. 915–940.

[24] S. Ma, "Alternating direction method of multipliers for sparse principal component analysis," *Journal of the Operations Research Society of China*, vol. 1, no. 2, pp. 253–274, Jun. 2013.

[25] L. Berk and D. Bertsimas, "Certifiably optimal sparse principal component analysis," *technical report*, 2016.

[26] R. Vershynin, *High-Dimensional Probability An Introduction with Applications in Data Science*. Draft, 2016.

[27] D. Bienstock, "Computational study of a family of mixed-integer quadratic programming problems," *Mathematical programming*, vol. 74, no. 2, pp. 121–140, 1996.

[28] A. Frangioni and C. Gentile, "SDP diagonalizations and perspective cuts for a class of nonseparable miqp," *Operations Research Letters*, vol. 35, no. 2, pp. 181–185, 2007.

[29] S. Burer and D. Vandenbussche, "Globally solving box-constrained nonconvex quad -ratic programs with semidefinite-based finite branch-and-bound," *Computational Optimization and Applications*, vol. 43, no. 2, pp. 181–195, 2009.

[30] P. Bonami, O. Günlük, and J. Linderoth, "Solving box-constrained nonconvex quadratic programs," *Optimization online*, pp. 26–76, 2016.

[31] S. S. Dey, A. M. Kazachkov, A. Lodi, and G. Munoz, "Cutting plane generation through sparse principal component analysis," 2021.

[32] A. Santana and S. S. Dey, "The convex hull of a quadratic constraint over a polytope," *SIAM Journal on Optimization*, vol. 30, no. 4, pp. 2983–2997, 2020.

[33] S. S. Dey, A. Santana, and Y. Wang, "New socp relaxation and branching rule for bipartite bilinear programs," *Optimization and Engineering*, vol. 20, no. 2, pp. 307–336, 2019.

[34] I. M. Bomze and G. Eichfelder, "Copositivity detection by difference-of-convex decomposition and ω-subdivision," *Mathematical Programming*, vol. 138, no. 1, pp. 365–400, 2013.

[35] S. Burer and A. Saxena, "Old wine in a new bottle: The milp road to miqcp," *Optimization Online*, 2009.

[36] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization. Interscience Series in Discrete Mathematics and Optimization*. 1988.

[37] J. Kim, "Cardinality constrained optimization problems," Ph.D. dissertation, Purdue University, West Lafayette, Indiana, Aug. 2016.

[38] X.-T. Yuan and T. Zhang, "Truncated power method for sparse eigenvalue problems," *arXiv preprint arXiv:1112.2679*, 2011.

[39] M. De Santis, F. Rendl, and A. Wiegele, "Using a factored dual in augmented lagrangian methods for semidefinite programming," *Operations Research Letters*, vol. 4 6, no. 5, pp. 523–528, 2018.

[40] S. S. Dey, M. Molinaro, and G. Wang, "Solving row-sparse principal component analysis via convex integer programs," *arXiv preprint arXiv:2010.11152*, 2020.

[41] V. Vu and J. Lei, "Minimax rates of estimation for sparse PCA in high dimensions," in *Artificial intelligence and statistics*, 2012, pp. 1278–1286.

[42] H. Attouch, J. Bolte, P. Redont, and A. Soubeyran, "Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the kurdyka-łojasiewicz inequality," *Mathematics of Operations Research*, vol. 35, no. 2, pp. 438–457, 2010.

[43] S. Ma, "Alternating direction method of multipliers for sparse principal component analysis," *Journal of the Operations Research Society of China*, vol. 1, no. 2, pp. 253–274, 2013.

[44] V. Q. Vu, J. Cho, J. Lei, and K. Rohe, "Fantope projection and selection: A near-optimal convex relaxation of sparse PCA," in *Advances in neural information processing systems*, 2013, pp. 2670–2678.

[45] J. Bolte, S. Sabach, and M. Teboulle, "Proximal alternating linearized minimization for nonconvex and nonsmooth problems," *Mathematical Programming*, vol. 146, no. 1-2, pp. 459–494, 2014.

[46] N. B. Erichson, P. Zheng, K. Manohar, S. L. Brunton, J. N. Kutz, and A. Y. Aravkin, "Sparse principal component analysis via variable projection," *SIAM Journal on Applied Mathematics*, vol. 80, no. 2, pp. 977–1002, 2020.

[47] S. Chen, S. Ma, L. Xue, and H. Zou, "An alternating manifold proximal gradient method for sparse PCA and sparse CCA," *arXiv preprint arXiv:1903.11576*, 2019.

[48] J. Kim, M. Tawarmalani, and J.-P. P. Richard, "Convexification of permutation-invariant sets and applications," *arXiv preprint arXiv:1910.02573*, 2019.

[49] Y. Li and W. Xie, "Exact and approximation algorithms for sparse pca," *arXiv preprint arXiv:2008.12438*, 2020.

[50] M. Asteris, D. Papailiopoulos, A. Kyrillidis, and A. G. Dimakis, "Sparse PCA via bipartite matchings," in *Advances in Neural Information Processing Systems*, 2015, pp. 766–774.

[51] A. Del Pia, *Sparse PCA on fixed-rank matrices*, 2019.

[52] C. D. Sigg and J. M. Buhmann, "Expectation-maximization for sparse and non-negative PCA," in *Proceedings of the 25th international conference on Machine learning*, ACM, 2008, pp. 960–967.

[53] L. W. Mackey, "Deflation methods for sparse PCA," in *Advances in neural information processing systems*, 2009, pp. 1017–1024.

[54] C. Boutsidis, P. Drineas, and M. Magdon-Ismail, "Sparse features for PCA-like linear regression," in *Advances in Neural Information Processing Systems*, 2011, pp. 2285–2293.

[55] M. Asteris, D. S. Papailiopoulos, and G. N. Karystinos, "Sparse principal component of a rank-deficient matrix," in *2011 IEEE International Symposium on Information Theory Proceedings*, IEEE, 2011, pp. 673–677.

[56] Q. Gu, Z. Wang, and H. Liu, "Sparse PCA with oracle property," in *Advances in neural information processing systems*, 2014, pp. 1529–1537.

[57] Y. Deshpande and A. Montanari, "Sparse PCA via covariance thresholding," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 4913–4953, 2016.

[58] R. Krauthgamer, B. Nadler, and D. Vilenchik, "Do semidefinite relaxations solve sparse PCA up to the information limit?" *The Annals of Statistics*, vol. 43, no. 3, pp. 1300–1322, 2015.

[59] Q. Berthet and P. Rigollet, "Computational lower bounds for sparse pca," *arXiv preprint arXiv:1304.0828*, 2013.

[60] T. Ma and A. Wigderson, "Sum-of-squares lower bounds for sparse pca," in *Advances in Neural Information Processing Systems*, 2015, pp. 1612–1620.

[61] T. T. Cai, Z. Ma, and Y. Wu, "Sparse PCA: Optimal rates and adaptive estimation," *The Annals of Statistics*, vol. 41, no. 6, pp. 3074–3110, 2013.

[62] Z. Wang, H. Lu, and H. Liu, "Tighten after relax: Minimax-optimal sparse PCA in polynomial time," in *Advances in neural information processing systems*, 2014, pp. 3383–3391.

[63] T. Cai, Z. Ma, and Y. Wu, "Optimal estimation and rank detection for sparse spiked covariance matrices," *Probability theory and related fields*, vol. 161, no. 3-4, pp. 781–815, 2015.

[64] J. Lei and V. Q. Vu, "Sparsistency and agnostic inference in sparse PCA," *The Annals of Statistics*, vol. 43, no. 1, pp. 299–322, 2015.

[65] A. Pietsch, *Operator ideals*. Deutscher Verlag der Wissenschaften, 1978, vol. 16.

[66] D. Steinberg, "Computation of matrix norms with applications to robust optimization," *Research thesis, Technion-Israel University of Technology*, vol. 2, 2005.

[67] L. A. Wolsey and G. L. Nemhauser, *Integer and combinatorial optimization*. John Wiley & Sons, 1999, vol. 55.

[68] G. Wang and S. S. Dey, "Upper bounds for model-free row-sparse principal component analysis," in *Proceedings of the International Conference on Machine Learning*, 2020.

[69] K. A. Gallivan and P. Absil, "Note on the convex hull of the stiefel manifold," *Technical note*, 2010.

[70] J.-B. Hiriart-Urruty and C. Lemaréchal, *Fundamentals of convex analysis*. Springer Science & Business Media, 2012.

[71] R. Kannan and S. Vempala, "Randomized algorithms in numerical linear algebra," *Acta Numerica*, vol. 26, p. 95, 2017.

[72] J. A. Tropp, "Column subset selection, matrix factorization, and eigenvalue optimization," in *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, SIAM, 2009, pp. 978–986.

[73] ——, "User-friendly tail bounds for sums of random matrices," *Foundations of computational mathematics*, vol. 12, no. 4, pp. 389–434, 2012.

[74] M. Mitzenmacher and E. Upfal, *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.

[75] S. O. Chan, D. Papailliopoulos, and A. Rubinstein, "On the approximability of sparse PCA," in *Conference on Learning Theory*, 2016, pp. 623–646.

[76] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine, "Broad patterns of gene expression revealed by clustering analysis of tumor

and normal colon tissues probed by oligonucleotide arrays," *Proceedings of the National Academy of Sciences*, vol. 96, no. 12, pp. 6745–6750, 1999.

[77] A. A. Alizadeh, M. B. Eisen, R. E. Davis, C. Ma, I. S. Lossos, A. Rosenwald, J. C. Boldrick, H. Sabet, T. Tran, X. Yu, *et al.*, "Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling," *Nature*, vol. 403, no. 6769, p. 503, 2000.

[78] S. S. Dey, G. Wang, and Y. Xie, "Approximation algorithms for training one-node relu neural networks," *IEEE Transactions on Signal Processing*, vol. 68, pp. 6696–6706, 2020.

[79] A. Toriello and J. P. Vielma, "Fitting piecewise linear continuous functions," *European Journal of Operational Research*, vol. 219, no. 1, pp. 86–95, 2012.

[80] A. Magnani and S. P. Boyd, "Convex piecewise-linear fitting," *Optimization and Engineering*, vol. 10, no. 1, pp. 1–17, 2009.

[81] Z. Yang, Z. Wang, H. Liu, Y. C. Eldar, and T. Zhang, "Sparse nonlinear regression: Parameter estimation under nonconvexity," *ICML'16 Proceedings of the 33rd International Conference on International Conference on Machine Learning*, vol. 48, pp. 2472–2481, 2016.

[82] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee, "Understanding deep neural networks with rectified linear units," *arXiv preprint arXiv:1611.01491*, 2016.

[83] M. Soltanolkotabi, "Learning ReLUs via gradient descent," in *Advances in Neural Information Processing Systems*, 2017, pp. 2007–2017.

[84] P. Manurangsi and D. Reichman, "The computational complexity of training relu (s)," *arXiv preprint arXiv:1810.04207*, 2018.

[85] S. M. M. Kalan, M. Soltanolkotabi, and A. S. Avestimehr, "Fitting relus via sgd and quantized SGD," *arXiv preprint arXiv:1901.06587*, 2019.

[86] S. Goel, V. Kanade, A. Klivans, and J. Thaler, "Reliably learning the relu in polynomial time," in *Conference on Learning Theory*, PMLR, 2017, pp. 1004–1042.

[87] L. Song, S. Vempala, J. Wilmes, and B. Xie, "On the complexity of learning neural networks," in *Advances in neural information processing systems*, 2017, pp. 5514–5522.

[88] A. Brutzkus and A. Globerson, "Globally optimal gradient descent for a convnet with gaussian inputs," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, pp. 605–614.

[89] S. S. Du, J. D. Lee, and Y. Tian, "When is a convolutional filter easy to learn?" *arXiv preprint arXiv:1709.06129*, 2017.

[90] S. S. Du, J. D. Lee, Y. Tian, B. Poczos, and A. Singh, "Gradient descent learns one-hidden-layer CNN: Don't be afraid of spurious local minima," *arXiv preprint arXiv:1712.00779*, 2017.

[91] S. Goel, V. Kanade, A. Klivans, and J. Thaler, "Reliably learning the ReLU in polynomial time," *arXiv preprint arXiv:1611.10258*, 2016.

[92] S. Goel, A. Klivans, and R. Meka, "Learning one convolutional layer with overlapping patches," *arXiv preprint arXiv:1802.02547*, 2018.

[93] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[94] S. M. Kakade, V. Kanade, O. Shamir, and A. Kalai, "Efficient learning of generalized linear and single index models with isotonic regression," in *Advances in Neural Information Processing Systems*, 2011, pp. 927–935.

[95] A. T. Kalai and R. Sastry, "The isotron algorithm: High-dimensional isotonic regression.," in *COLT*, Citeseer, 2009.

[96] R. Livni, S. Shalev-Shwartz, and O. Shamir, "On the computational efficiency of training neural networks," in *Advances in neural information processing systems*, 2014, pp. 855–863.

[97] S. Mei, Y. Bai, and A. Montanari, "The landscape of empirical risk for nonconvex losses," *The Annals of Statistics*, vol. 46, no. 6, pp. 2747–2774, 2018.

[98] M. Raginsky, A. Rakhlin, and M. Telgarsky., "Non-convex learning via stochastic gradient langevin dynamics: A nonasymptotic analysis," in *Conference on Learning Theory (COLT)*, 2017.

[99] X. Zhang, Y. Yu, L. Wang, and Q. Gu, "Learning one-hidden-layer relu networks via gradient descent," in *The 22nd International Conference on Artificial Intelligence and Statistics*, PMLR, 2019, pp. 1524–1534.

[100] Y. Cao and Q. Gu, "Tight sample complexity of learning one-hidden-layer convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 10 612–10 622.

[101] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin, "The elements of statistical learning: Data mining, inference and prediction," *The Mathematical Intelligencer*, vol. 27, no. 2, pp. 83–85, 2005.

[102] D. Bertsimas, A. King, and R. Mazumder, "Best subset selection via a modern optimization lens," *The annals of statistics*, vol. 44, no. 2, pp. 813–852, 2016.

[103] 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.

[104] D. Boob, S. S. Dey, and G. Lan, "Complexity of training ReLU neural network," *arXiv preprint arXiv:1809.10787*, 2018.

[105] M. Mickey, P. Mundle, D. Walker, and A. Glinsk, "Test criteria for Pearson type III distributions.," CEIR INC BEVERLY HILLS CALIF, Tech. Rep., 1963.

[106] R. I. Jennrich, "Asymptotic properties of non-linear least squares estimators," *The Annals of Mathematical Statistics*, vol. 40, no. 2, pp. 633–643, 1969.

[107] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *arXiv preprint arXiv:1611.03530*, 2016.

[108] S. Oymak, B. Recht, and M. Soltanolkotabi, "Sharp time–data tradeoffs for linear inverse problems," *IEEE Transactions on Information Theory*, vol. 64, no. 6, pp. 4129–4158, 2017.

[109] T. Laurent and J. Brecht, "The multilinear structure of relu networks," in *International Conference on Machine Learning*, PMLR, 2018, pp. 2908–2916.

[110] G. Wang, G. B. Giannakis, and J. Chen, "Learning relu networks on linearly separable data: Algorithm, optimality, and generalization," *IEEE Transactions on Signal Processing*, vol. 67, no. 9, pp. 2357–2370, 2019.

[111] I. Diakonikolas, S. Goel, S. Karmalkar, A. R. Klivans, and M. Soltanolkotabi, "Approximation schemes for relu regression," *arXiv preprint arXiv:2005.12844*, 2020.

[112] S. Frei, Y. Cao, and Q. Gu, "Agnostic learning of a single neuron with gradient descent," *arXiv preprint arXiv:2005.14426*, 2020.

[113] J. Jeffers, "Two case studies in the application of principal component analysis," *Applied Statistics*, pp. 225–236, 1967.

[114] A. d'Aspremont, F. R. Bach, and L. E. Ghaoui, "Full regularization path for sparse principal component analysis," in *Proceedings of the 24th international conference on Machine learning*, ACM, 2007, pp. 177–184.

[115] S. Bagroy, P. Kumaraguru, and M. De Choudhury, "A social media based index of mental well-being in college campuses," in *Proceedings of the 2017 CHI Con-*

ference on Human Factors in Computing Systems, ser. CHI '17, Denver, Colorado, USA: ACM, 2017, pp. 1634–1646, ISBN: 978-1-4503-4655-9.

[116] K. Saha and M. De Choudhury, "Modeling stress with social media around incidents of gun violence on college campuses," *Proc. ACM Hum.-Comput. Interact.*, vol. 1, no. CSCW, 92:1–92:27, Dec. 2017.

[117] J. W. Pennebaker, M. E. Francis, and R. J. Booth, "Linguistic inquiry and word count: LIWC 2001," *Mahway: Lawrence Erlbaum Associates*, vol. 71, no. 2001, p. 2001, 2001.

[118] Y. R. Tausczik and J. W. Pennebaker, "The psychological meaning of words: LIWC and computerized text analysis methods," *Journal of language and social psychology*, vol. 29, no. 1, pp. 24–54, 2010.

[119] S. Burer and R. D. Monteiro, "Local minima and convergence in low-rank semidefinite programming," *Mathematical Programming*, vol. 103, no. 3, pp. 427–444, 2005.