ALGORITHMS FOR SPARSITY CONSTRAINED PRINCIPAL COMPONENT ANALYSIS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

By
Fatih Selim Aktaş
July 2023

Algorithms for Sparsity Constrained Principal Component Analysis
By Fatih Selim Aktaş
July 2023

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Mustafa Çelebi Pınar(Advisor)

Firdevs Ulus

Cem İyigün

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

ALGORITHMS FOR SPARSITY CONSTRAINED PRINCIPAL COMPONENT ANALYSIS

Fatih Selim Aktaş
M.S. in Industrial Engineering
Advisor: Mustafa Çelebi Pınar
July 2023

The classical Principal Component Analysis problem consists of finding a linear transform that reduces the dimensionality of the original dataset while keeping most of the variation. Extra sparsity constraint sets most of the coefficients to zero which makes interpretation of the linear transform easier. We present two approaches to the sparsity constrained Principal Component Analysis.

Firstly, we develop computationally cheap heuristics that can be deployed in very high-dimensional problems. Our heuristics are justified with linear algebra approximations and theoretical guarantees. Furthermore, we strengthen our algorithms by deploying the necessary conditions for the optimization model.

Secondly, we use a non-convex log-sum penalty in the semidefinite space. We show a connection to the cardinality function and develop an algorithm, PCA Sparsified, to solve the problem locally via solving a sequence of convex optimization problems. We analyze the theoretical properties of this algorithm and comment on the numerical implementation. Moreover, we derive a pre-processing method that can be used with previous approaches.

Finally, our findings from the numerical experiments we conducted show that our greedy algorithms scale to high dimensional problems easily while being highly competitive in many problems with state-of-art algorithms and even beating them uniformly in some cases. Additionally, we illustrate the effectiveness of PCA Sparsified on small dimensional problems in terms of variance explained. Although it is computationally very demanding, it consistently outperforms local and greedy approaches.

Keywords: Sparse PCA, Greedy Algorithms, SDP.

ÖZET

SEYREK KISITLI TEMEL BİLEŞEN ANALİZİ İÇİN ALGORİTMALAR

Fatih Selim Aktaş
Endüstri Mühendisliği, Yüksek Lisans
Tez Danışmanı: Mustafa Çelebi Pınar
Temmuz 2023

Klasik Temel Bileşen Analizi problemi, orijinal veri kümesinin boyutunu azaltırken varyansyonun çoğunluğunu koruyan bir lineer dönüşüm bulmayı hedefler. Ekstra seyreklik kısıtı, katsayıların çoğunu sıfıra sabitler ve lineer dönüşümün yorumlanmasını kolaylaştırır. Seyreklik kısıtlı Temel Bileşen Analizi için iki yaklaşım sunuyoruz.

İlk olarak, çok yüksek boyutlu problemlerde kullanılabilecek hesaplamalı olarak ucuz sezgisel yöntemler geliştiriyoruz. Sezgisel yöntemlerimiz, doğrusal cebir yaklaşımları ve teorik garantilerle haklı çıkarılmıştır. Ayrıca, optimizasyon modeli için gerekli koşulları uygulayarak algoritmalarımızı güçlendiriyoruz.

İkincisi, yarı belirli uzayda dışbükey olmayan log-toplam cezası kullanıyoruz. Bu fonksiyonun kardinalite fonksiyonuyla bağlantısını gösteriyor ve bu problemi çözmek için PCA Sparsified adlı bir dizi dışbükey optimizasyon problemini çözen bir algoritma geliştiriyoruz. Bu algoritmanın teorik özelliklerini analiz ediyor ve sayısal uygulaması hakkında yorum yapıyoruz. Ayrıca, önceki yaklaşımlarla birlikte kullanılabilecek bir ön işleme yöntemi türetiyoruz.

Son olarak, yaptığımız sayısal deneylerden elde ettiğimiz bulgular, açgözlü algoritmalarımızın yüksek boyutlu problemlere kolayca ölçeklendiğini ve birçok problemde halihazırda mevcut olan algoritmalarla oldukça rekabetçi olduğunu hatta bazı durumlarda devamlı olarak geride bıraktığını göstermektedir. Ayrıca, PCA Sparsified'in açıklanan varyans açısından düşük boyutlu problemlerdeki etkinliğini gösteriyoruz. Hesapsal olarak pahalı olmasına rağmen, yerel ve açgözlü yaklaşımları sürekli olarak geride bırakmaktadır.

Anahtar sözcükler: Seyrek TBA, Açgözlü Algoritmalar, PYBOP.

Acknowledgement

I would like to express my deepest gratitude to all those who have contributed to the completion of this Master's thesis.

First and foremost, I am sincerely grateful to my thesis advisor, Mustafa Pınar, for his unwavering support and invaluable guidance in shaping my life. I was fortunate to meet and work with him. I can say without a shred of doubt that he was the greatest mentor I ever had.

I want to also recognize the great advice and leadership of Çağın Ararat, Emin Karagözoğlu, Alper Şen, and Sinan Gezici during my studies.

I extend my heartfelt thanks to the members of my thesis committee, Firdevs Ulus and Cem İyigün for their helpful suggestions and time.

I am deeply indebted to my friends who made this journey bearable. I wish to express my heartfelt appreciation to Barlas Tarık Arslan, Orhan Uysal, Selçuk Balcıoğlu, Bora Çetin, Enes Şamil, Ömer Ekmekcioğlu, Deniz Akkaya, Yiğit Ateş, Doruk Karakaş, and Ömer Durukan Kılıç.

I want to also thank TÜBİTAK for their support during my Master's Degree studies.

Finally, I would like to acknowledge the support and encouragement from my family. I devote this thesis to my mother, who calmed my howling mind and reassured me even when I was in nothing but despair and doubt.

Contents

1 Introduction **1**

1.1 Problem Definition and Motivation 2

1.2 Challenges 3

1.3 Applications 4

2 Literature Review **5**

2.1 Local Models 5

2.1.1 Gradient Algorithms 5

2.1.2 Greedy Algorithms 6

2.2 Semidefinite Relaxations 6

2.3 Exact Approaches 7

3 Improved Greedy Enumeration Algorithms **8**

3.1 Notation 8

3.2 Eigenvalue Bounds 9

3.2.1 Gershgorin Disc 9
3.2.2 Frobenius Norm 10
3.2.3 QR Factorization and Correlations 11
3.3 Algorithms 12
3.3.1 Algorithm Inspired by Correlations 13
3.3.2 Algorithm Inspired by Gershgorin Discs 14
3.3.3 Algorithm Inspired by Frobenius Norm 15
3.3.4 Computational Complexity 16
3.3.5 Shrinking the Search Space 17
3.4 Enforcing Necessary Conditions 17
3.4.1 Stationarity 18
3.4.2 CW-Optimality 18
3.4.3 Enhancing the Solutions via Necessary Conditions 19
3.5 Multiple Components and Deflation 20

4 PCA Sparsified **21**

4.1 Notation 21
4.2 New Perspective for SPCA 22
4.3 Semidefinite Relaxation 23
4.3.1 Reweighted Optimization 24

4.3.2 Connection to DSPCA 26

4.4 Algorithms for the Subproblem of PCA Sparsified 28

4.4.1 ADMM 28

4.4.2 CGAL 33

4.4.3 ADMM vs CGAL 36

4.5 Analysis of PCA Sparsified 38

4.5.1 Convergence in Objective Function Value 38

4.5.2 Stationarity of Limit Points 40

4.5.3 Frank-Wolfe Gap Convergence 41

4.5.4 Dual Bounds 43

4.6 Multiple Components 45

4.7 Sufficient Conditions for Theoretical Guarantees 47

4.7.1 Local Minimum and Rank-one Connection 47

4.7.2 Strict Local minimum of the surrogate function 49

4.8 Implementation and Computational Tactics 51

5 Numerical Experiments **54**

5.1 Scalable Algorithms 54

5.1.1 Synthetic Data Tests 56

5.1.2 Random Datasets 58

5.1.3 Real Datasets 59

5.1.4 Multiple Components 62

5.2 Small Dimensional Problems 63

5.2.1 Random Data 64

5.2.2 Computation Times 66

5.2.3 Variance Throughout Outer Iterations 66

5.2.4 Results on Real Data 67

6 Conclusion **72**

A Sufficient Condition for Convergence of ADMM **83**

B ADMM Residual Calculations **84**

C CGAL Smoothing Details **86**

D Proof of convergence in distance between consecutive X **88**

E Pre-processing for DSPCA **91**

List of Figures

4.1 Log-sum, ℓ_1, and ℓ_0 penalty 27

5.1 Run Time and Variance Explained on Synthetic Datasets 57

5.2 Run Time and Variance Explained on Random Datasets 58

5.3 Variance Explained on Pitprops Dataset 60

5.4 Run Time and Variance Explained on AT & T and Arcene Datasets 61

5.5 Run Time and Variance Explained on AT & T and Arcene Datasets for the Multiple Components Problem 69

5.6 Small Dimensional Random Data with $n = 200$ 70

5.7 Small Dimensional Random Data with $n = 400$ 70

5.8 Variance and Computation Times Against Problem Size 70

5.9 Sparsifying Effect of Reweighted Optimization 71

5.10 Variance Explained on Arrhythmia and FMA Datasets 71

List of Tables

3.1 Computational Complexity of the Proposed Greedy Algorithms . 16

5.1 Exact Recovery Rates of Different Algorithms for Varying Sparsity Levels from 2 to 12 on Pitprops Dataset 59

5.2 Illustration of effect of choice of stability parameter for the underdetermined dataset with feature size $n = 400$ 65

Chapter 1

Introduction

This thesis presents two approaches to the sparsity (or cardinality) constrained Principle Component Analysis (PCA) problem. The goal of solving this problem is to find linear combinations of features of a data matrix that maximizes the variance. This approach allows the reduction of the dimension of the problem with minimal loss of the data. While this is closely related to the original PCA problem, sparsity constraint limits the number of non-zero variables in the linear combinations computed. With a small number of non-zero factors, interpretation of them becomes easier which might be insightful in physical or biological applications. Indeed some of the successful applications of Sparse PCA are genetics, [1, 2, 3], face recognition, [4] and signal processing, [5].

In the first method, we utilize eigenvalue approximation ideas such as the Gershgorin Circle Theorem to come up with multiple novel heuristics. Using previous results from the literature, we show approximation guarantees for these algorithms. Furthermore, we employ necessary optimality conditions previously studied in the literature to enhance the quality of the solutions computed.

In the second method, we develop a new algorithm built upon convex relaxations using Semidefinite Programming (SDP) for the Sparse PCA problem. Then, we take advantage of reweighted optimization to compute sparse solutions.

Thus, our algorithm iteratively solves a convex optimization problem with new parameters at each iteration. We show theoretical guarantees for this approach and propose two numerical algorithms. Moreover, we develop a pre-processing method that can be also used with previously proposed solution methods.

1.1 Problem Definition and Motivation

Principal Component Analysis is a classical statistical method used for data compression, visualization, and clustering. PCA has a large number of applications in machine learning, statistics, finance, genetics, signal processing, and many other science and engineering problems. Given a data set, PCA amounts to finding linear factors that maximize variance. Let $B \in \mathbb{R}^{m \times n}$ be the data matrix comprised of m observations of n features. Assuming B is centered[1], PCA can be written as the following optimization problem using the variance/covariance matrix $A = B^T B$

$$\begin{aligned} \max_{x} \quad & x^T A x \\ \text{st.} \quad & ||x||_2 = 1. \end{aligned} \tag{1.1}$$

Solutions to this problem are referred to as loadings. The projection of the data matrix onto the loadings is called the first principal component. The problem is solved by computing the dominant eigenvector of A. Extracting multiple components can be done iteratively using deflation methods or they can be computed concurrently using subspace iteration methods. PCA is often used to reduce the dimension of data matrix B by projecting it onto the eigenvectors $v_1(A), \ldots, v_p(A)$, where $v_i(A)$ is the eigenvector corresponding to ith largest eigenvalue of A. As a consequence, loadings are orthogonal and components are uncorrelated.

The main drawback of PCA is that computed loadings are often dense, i.e., there are only a few zero-valued variables in the loading vectors. Interpretation of such dense loadings is usually difficult as components are constructed by taking

[1] A centered data matrix is obtained by subtracting the mean value of each column from every entry of that column.

a linear combination of all variables. In many applications, the data matrix is often high dimensional. For example, gene expression data often consist of a small number of observations of a large number of genes. Using PCA a gene can be expressed in a lower dimensional basis. However, since each term will be a linear combination of all terms, it is unclear what each of the principal components represents. Principal components that are constructed by sparse loadings would be much easier to interpret. Another instance where the sparsity of loadings might be useful is financial applications where there could be fixed transaction costs incurred for nonzero portfolio positions. Hence, sparsity in loadings is often desirable. These disadvantages of PCA lead to the consideration of the following modified problem:

$$\begin{aligned} \max_{x} \quad & x^T A x \\ \text{st.} \quad & ||x||_2 = 1 \\ & ||x||_0 \leq s, \end{aligned} \tag{1.2}$$

where $||x||_0$ denotes the cardinality of x, i.e., it counts the number of nonzero elements of x, and s is the parameter controlling its sparsity (cardinality) level. It is clear that the objective function value of this problem is upper bounded by that of PCA. Thus, the cardinality constraint controls the trade-off between statistical fidelity (variance explained) and interpretability (sparse loadings).

1.2 Challenges

Unlike PCA, Sparse PCA (henceforth, SPCA) is a difficult problem. It was shown in [6] that SPCA is NP-hard by reducing it to the sparsity constrained least squares problem. Furthermore, unlike in PCA, extracting multiple components using SPCA is not trivial. Moreover, the orthogonality of loadings could be lost, and components could be correlated. Despite these difficulties, SPCA has been a hot research topic for the last few decades, as we discuss in chapter 2.

1.3 Applications

Sparsity constrained optimization has numerous applications in finance [7] for finding portfolios with a limited number of non-zero portfolio positions (thus leading to smaller transaction costs), and in the field of Compressed Sensing and subset selection [8], where the so-called RIP constant plays an important role for the recovery of a sparse signal using convex optimization approaches. The effectiveness of SPCA was shown for different tasks in [9] for isolating relevant genes in gene expression data, and in [10] for summarizing large text data sets.

The rest of this thesis includes a literature review of the SPCA problem that comprised of previously proposed models, theoretical analysis, and computational approaches. After that, we introduce two different approaches to the SPCA problem developed in the Improved Greedy Enumeration Algorithms for Sparse PCA and PCA Sparsified studies.

Chapter 2

Literature Review

Given the importance of SPCA, several approaches were developed to solve the SPCA problem which we categorize under three classes: local models of SPCA that efficiently compute high-quality solutions, convex relaxations of SPCA that compute near-optimal solutions and give upper bounds on the original problem, and finally exact approaches that provably solve SPCA by combinatorial search.

2.1 Local Models

Given the nature of computations involved in estimating sparse loadings, we can further classify local models under two umbrellas; algorithms use gradient-based local optimization and other greedy-type local optimization algorithms.

2.1.1 Gradient Algorithms

The simplest approach to SPCA using a thresholding method was proposed in [11]. However, this approach often gives poor results in practice. A LASSO-based approach, namely ℓ_1 norm penalized PCA, was proposed in [12]. However, this

method does not result in a convex optimization problem, therefore this approach proved computationally expensive. This approach was extended in [13] using elastic-net which uses both ℓ_1 and ℓ_2 norm penalty that preserves convexity. A computationally more appealing non-convex optimization approach was proposed in [14] using a Newton-type algorithm with Hessian approximation.

In [15], an efficient method to compute sparse principal components based on expectation-maximization algorithm was given. Similar work was done in [16] using a gradient framework (named generalized power method) which extended the work to SPCA over different manifolds (block SPCA, Stiefel Manifold). Later, these algorithms were recognized to be special cases of the conditional gradient algorithm [17]. More recently, efficient algorithms were developed in [18] using alternating maximization method which was shown to be equivalent to the generalized power method.

2.1.2 Greedy Algorithms

Greedy forward selection was applied to the SPCA problem in [19]. However, greedy forward selection is a computationally expensive method due to eigenvalue computations in each iteration. A computationally less expensive algorithm with similar performance was given in [8], referred to as the approximate greedy search algorithm. Moreover, in [8], the authors also derived sufficient conditions for optimality for the penalized version of SPCA. Similarly, in [20] previously proposed greedy algorithms were reinterpreted as Coordinate-wise Ascent algorithms that converge to a point satisfying a stronger type of optimality condition than were the case for Conditional Gradient (CG)-type algorithms.

2.2 Semidefinite Relaxations

The first contribution based on a convex semidefinite relaxation to the SPCA problem was proposed by d'Aspremont *et al.* [21] using a semidefinite relaxation

called DSPCA. In [22], the same relaxation problem was derived using Lagrangian duality, and an efficient algorithm was developed to solve the dual of the problem. A more specialized algorithm for the DSPCA model was proposed in [10]. Moreover, these authors also apply a pre-processing method derived in [8] for the sparsity penalized model to their convex relaxation to reduce the dimension of the problem.

Semidefinite relaxation methods were also extended to computing multiple sparse components by solving a single optimization model instead of relying on deflation methods. In [23], the Stiefel manifold with additional constraints is used to model this problem at the cost of loss of convexity. A convex relaxation of the Stiefel manifold (Fantope) was used in [24] for the same problem.

Recovery aspects of SPCA algorithms also received some attention. Recovery of sparse components using DSPCA and diagonal thresholding for spiked covariance models were studied in [25]. More general recovery and estimation bounds were derived in [26, 27] for subspaces, i.e., multiple components.

2.3 Exact Approaches

Methods to provably solve the SPCA problem require a type of search algorithm. Moghaddam *et al.* [19] propose a branch and bound algorithm that can handle small-scale problems. This approach was refined by [28] to handle orthogonality and correlations for the multiple components problem. More scalable approaches were developed in [29, 30] using branch and bound and cutting plane methods, respectively. More involved mixed integer programming models for the exact solution using linear programming or semidefinite programming are discussed in [31, 32].

Chapter 3

Improved Greedy Enumeration Algorithms

In this chapter, we develop multiple heuristics that rely on eigenvalue bounds. Our proposed algorithms use these heuristics to construct subsets that have large variances. Furthermore, we apply a first-order necessary condition to increase the variance explained.

3.1 Notation

$\mathbf{S}^n$ and $\mathbf{S}^n_+$ denote set of symmetric matrices of size n, and positive semi-definite symmetric matrices of size n, respectively. We denote by $\lambda_i(X)$ and $v_i(X)$ the i'th eigenvalue and corresponding i'th eigenvector of X, in descending order of eigenvalues. For some index set C and matrix X, X_C denotes the sub-matrix of X induced by C. $||X||_F$ denotes the Frobenius norm of X. $\text{tr}(X)$ denotes trace of matrix X.

3.2 Eigenvalue Bounds

In this section, we describe the main ingredients used in our algorithms for eigenvalue approximation. Eigenvalue bounds are used in theoretical analysis of the algorithms as done in [33, 34], and deployed as a tool in heuristics to avoid solving the eigenvalue problem completely.

3.2.1 Gershgorin Disc

Gershgorin Circle Theorem [35] states that every eigenvalue of a matrix A lies within at least one of Gershgorin discs $D(a_{ii}, R_i) \subseteq \mathbb{C}$ where $D(a_{ii}, R_i)$ is a disc with center a_{ii} and radius R_i which are defined as follows

$$\begin{aligned} a_{ii} &= e_i^T A e_i \\ R_i &= \textstyle\sum_{j \neq i} |a_{ij}| \end{aligned} \tag{3.1}$$

where e_i is the i'th column of identity matrix. A special case of the theorem is obtained when the matrix A is symmetric since symmetric matrices have real eigenvalues. Then, the Gershgorin disc reduces to a Gershgorin interval. Let $I = [l_i, u_i]$ denote the Gershgorin interval corresponding to Gershgorin disc $D(a_{ii}, R_i)$. Then one has

$$\begin{aligned} l_i &= a_{ii} - R_i \\ u_i &= a_{ii} + R_i. \end{aligned} \tag{3.2}$$

Moreover, this interval can even be sharpened for our problem as the covariance matrix A is also at least symmetric positive semidefinite. Then one gets

$$\begin{aligned} l_i &= \max(a_{ii} - R_i, 0) \\ u_i &= a_{ii} + R_i. \end{aligned} \tag{3.3}$$

The u_i values give an upper bound on the dominant eigenvalue. Then, applying the upper bound on the sparse eigenvalue problem we obtain

$$p \leq u_i = \textstyle\sum_{j \in S} |a_{ij}|, \tag{3.4}$$

where S is some sparsity pattern or index set computed and p is the optimal value. This bound motivates the following idea for finding the sparse dominant eigenvalue of the matrix A: construct the largest possible Gerschgorin disc for each variable. Furthermore, in [33] it was shown that an algorithm based on this idea has $1/\sqrt{s}$ worst-case approximation ratio. In other words, let q be the value of the solution that Algorithm 2 described in section 3.3 returns. Then the following inequality holds

$$q\sqrt{s} \geq p. \tag{3.5}$$

Similar to Gershgorin Discs, one can use Brauer's Cassini ovals [36] which locate eigenvalues using the following inequality:

$$|\lambda - a_{ii}||\lambda - a_{jj}| \leq R_i R_j, \tag{3.6}$$

where a_{ii} and R_i are as defined before, and λ denotes any eigenvalue. All eigenvalues lie in the union of all Brauer Cassini ovals. This fact furnishes another upper bound on the dominant eigenvalue and is often sharper than Gershgorin Discs. However, unlike Gershgorin Discs where every Gershgorin Disc has at least one eigenvalue, not all Brauer's Cassini ovals contain an eigenvalue. This is important because our algorithms construct sparsity patterns that give the best upper bounds. Hence, constructing sparsity patterns using Brauer's Cassini ovals as an approximation to the dominant eigenvalue does not yield good sparsity patterns in practice.

3.2.2 Frobenius Norm

Recall that the Frobenius norm of a matrix is given as

$$||A||_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^2 = tr(A^T A) = \sum_{i=1}^{n} \lambda_i^2, \tag{3.7}$$

where λ_i is the i'th eigenvalue of A. Since $\lambda_1 \geq \lambda_j \geq 0,\ \forall j$, we get the following inequality

$$\lambda_1^2 \leq ||A||_F^2. \tag{3.8}$$

The above inequality holds as equality if $\lambda_2 = 0$, hence all eigenvalues are zero except the dominant eigenvalue in that case. Similarly, we can write the following inequality which holds as equality when all eigenvalues are equal;

$$\frac{||A||_F^2}{n} \leq \lambda_1^2. \tag{3.9}$$

A similar bound could be derived using matrix trace. In our setting we assume that the matrix is normalized, i.e., all diagonal entries are equal. Therefore, trace inequality does not give any further information.

In general, the main drawback of this approximation is that finding the sub-matrix with the largest Frobenius norm is also NP-Hard due to its combinatorial nature[1]. Moreover, the sub-matrix with the largest eigenvalue does not necessarily coincide with the sub-matrix with the largest Frobenius norm. That being said, the Frobenius norm is still a good approximation since it gives both a lower and an upper bound on the dominant eigenvalue, and it is computationally cheaper when used in greedy forward selection algorithms compared to eigenvalue computation.

3.2.3 QR Factorization and Correlations

The most intuitive bound can be derived using matrix norms. An upper bound for the objective function could be found using the Frobenius norm or matrix

[1]The problem of computing a maximum cardinality clique in a graph, which is a well-known NP-complete problem [37], can be polynomially reduced to this problem. However, we do not go into the details here.

ℓ_1-norm. In particular, we have

$$\begin{aligned} x^T Ax &= \text{tr}(Axx^T) \\ \text{tr}(Axx^T) &\leq ||A||_1 ||xx^T||_\infty \\ &\leq ||A||_1, \end{aligned} \tag{3.10}$$

where $||A||_1 = \sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}|$ and $||A||_\infty = \max_{i,j} |a_{ij}|$. The second line follows from Hölder's Inequality, and the last line follows from the scale constraint $||x||_2 \leq 1$.

We can also use the QR decomposition of A to realize that if the correlation between the variables is high, then the dominant eigenvalue of the matrix is also large [38]. Since we have assumed that the diagonal entries of the matrix A are equal, any sub-matrix of the same size have an equal trace, i.e., have an equal sum of eigenvalues. Furthermore, as the data matrix is a covariance matrix, it is symmetric and positive semi-definite, i.e., all eigenvalues are non-negative. When we approximate the eigen-decomposition of the matrix A using QR decomposition, as in the first step of the QR algorithm, if the columns are highly correlated, the norm of the first column R_{11} will be large and the rest of the diagonal entries of R matrix will be small. This will be very important when selecting sub-matrices to handle the sparsity constraint since the sum of the diagonal elements should have an equal sum for each identically sized sub-matrix, and we can find large dominant eigenvalues from the sub-matrix generated by selecting the correlated columns.

3.3 Algorithms

The proposed algorithms are based on approximations of sparse eigenvalues and greedy selection of these eigenvalues using a forward selection approach. This allows us to find strong candidates when the sparsity level is high, i.e., the value of s in the cardinality constraint is small. Checking eigenvalue approximations

enables us to effectively use the greedy selection methods as this approach is computationally tractable compared to calculating eigenvalues in our combinatorial setting.

The algorithms presented below are similar in the sense that they perform two steps to find the solution. The first step consists of the eigenvalue approximation using one of the approximation methods described in section 2. This step gives us the set of candidate indices that give high eigenvalues when selected. The second step calculates the eigenvalues corresponding to the sub-matrix selected from the sparse indices found in step 1 and decides on the eigenvalue-eigenvector pair. The main difference between our algorithm and its competitors such as [33] and [34] is the two-step approach we follow. In the aforementioned studies, a hard-thresholding method is used either on the columns of the data matrix or in the eigenvectors to obtain sparse solutions. Different from this approach, we proceed with the indices obtained from the hard-thresholding to create multiple sub-matrices with entries corresponding to the indices found in each column. In our second step, we solve the problem for every sub-matrix and select the best solution among the candidates.

3.3.1 Algorithm Inspired by Correlations

Analytical justification for Algorithm 1 can be stated as follows; if we can select the largest ℓ_1 norm submatrix, we can guarantee an approximation ratio of $1/(s\sqrt{s})$ by using uniform norm bounds. Let S be the subset that gives the largest eigenvalue and C be the subset gives largest ℓ_1 norm submatrix. The following inequalities yield the approximation ratio

$$\begin{aligned} ||A_S||_2 \leq ||A_S||_1 \leq ||A_C||_1 \\ ||A_C||_1 \leq s\sqrt{s}||A_C||_2. \end{aligned} \tag{3.11}$$

In the first and the last inequalities we use the uniform norm bounds between the spectral norm and the ℓ_1 norm of the matrix A in both directions, i.e., $||A||_2 \leq ||A||_1$ and $||A||_1 \leq s\sqrt{s}||A||_2$ [39]. The second inequality follows from optimality

Algorithm 1: Sparse PCA Algorithm based on ℓ_1 norm (CCW)

Input: Covariance Matrix $A \in \mathbf{S}_+^n$, sparsity level s
Set $\lambda_{CCW} = 0, v_{CCW} = \emptyset, CCW = \emptyset$
for $i = 1, 2, \ldots, n$ **do**
  Set $C = \{i\}$
  while $|C| < s$ **do**
    Compute index $j = \arg\max_k \left(\|A_{C\cup\{k\}}\|_1\right)$
    Update $C = C \cup \{j\}$.
  end while
  Compute dominant eigenpair $\lambda_1(A_C), v_1(A_C)$
  if $\lambda_{CCW} < \lambda_1(A_C)$ **then**
    Set $CCW = C$
    Set $\lambda_{CCW} = \lambda_1(A_C)$
    Set $v_{CCW} = v_1(A_C)$
  end if
end for
return $\lambda_{CCW}, v_{CCW}, CCW$

of subset C.

The approximation ratio derived in (3.11) is not sharp since the inequality $||A||_2 \leq ||A||_1$ holds as an equality when A has at most one non-zero entry, and the inequality $||A||_1 \leq s\sqrt{s}||A||_2$ holds as an equality for Fourier matrices which are not positive semi definite. The largest value that we were able to compute for the optimization problem $\max\{||A||_1; ||A||_2 \leq 1, A \succeq 0\}$ is equal to $2s - 2$ which is attained for $A = I_s - \frac{1}{s}\mathbf{1}\mathbf{1}^T$. Even then, the bound $2s - 2$ cannot be achieved by the algorithm as the sharp bound would require comparison of two subsets that satisfy $||A||_2 \leq ||A||_1$ and $||A||_1 \leq (2s - 2)||A||_2$ as equality. However, the first inequality is sharp for matrices with at most a single non-zero entry, which clearly can be improved in the ℓ_1 norm sense.

3.3.2 Algorithm Inspired by Gershgorin Discs

As alluded to in the previous section, following the analysis in [33] Algorithm 2 achieves a worst-case approximation ratio equal to $1/s$.

Algorithm 2: Sparse PCA Algorithm based on Gershgorin Discs (GD)

1: **Input:** Covariance Matrix $A \in \mathbf{S}_+^n$, sparsity level s
2: Set $\lambda_{GD} = 0, v_{GD} = \emptyset, GD = \emptyset$
3: For $i = 1, 2, \ldots, n$, find S that maximizes $||e_i^T A_S||_1$ such that $S \subseteq \{1, 2, \ldots, n\}$, $|S| = s$, i.e., find the indices of the s largest values in row i. Store the indices found from all of the rows in a matrix $C \in \mathbb{R}^{n \times s}$ to index the important features for each row.
4: For all $i = 1, \ldots, n$, find the dominant eigenvalue of sub-matrix $\lambda_1(A_{C_i})$, and the corresponding eigenvector $v_1(A_{C_i})$, where C_i is the i'th row of matrix C.
5: Compute $j = \arg\max_i \lambda_1(A_{C_i})$
6: Set $GD = C_j$
7: Set $\lambda_{GD} = \lambda_1(A_{C_j})$
8: Set $v_{GD} = v_1(A_{C_j})$
9: **return** λ_{GD}, v_{GD}, GD

3.3.3 Algorithm Inspired by Frobenius Norm

Similar to the approximation ratio given in (3.5) for Algorithm 2, in [34], a $1/\sqrt{s}$ approximation ratio is proved for a similar algorithm. This result motivates Algorithm 3. Let S be the optimal subset that has maximal eigenvalue and FR be the subset that has maximal Frobenius norm. The following inequality is satisfied

$$||A_{FR}||_2 \sqrt{s} \geq ||A_S||_2. \tag{3.12}$$

Our algorithms can be recognized as modified versions of greedy forward selection proposed by Moghaddam *et al.* [19]. However, instead of finding the dominant eigenvalue at each iteration, we use the heuristics derived in Section 3.2 for the dominant eigenvalue and mathematically argue why these algorithms work. Some of these heuristics are also utilized in the algorithms proposed in [29, 33, 34]. This allows us to quickly construct good sparsity patterns.

Algorithm 3: Sparse PCA Algorithm based on Frobenius norm (FCW)

Input: Covariance Matrix $A \in \mathbf{S}_+^n$, sparsity level s
Set $\lambda_{FCW} = 0, v_{FCW} = \emptyset, FCW = \emptyset$
for $i = 1, 2, \ldots, n$ **do**
  Set $C = \{i\}$
  while $|C| < s$ **do**
    Compute index $j = \arg\max_k \left(\|A_{C\cup\{k\}}\|_F\right)$
    Update $C = C \cup \{j\}$.
  end while
  Compute dominant eigenpair $\lambda_1(A_C), v_1(A_C)$
  if $\lambda_{FCW} < \lambda_1(A_C)$ **then**
    Set $FCW = C$
    Set $\lambda_{FCW} = \lambda_1(A_C)$
    Set $v_{FCW} = v_1(A_C)$
  end if
end for
return $\lambda_{FCW}, v_{FCW}, FCW$

3.3.4 Computational Complexity

The computational complexity of computing good sparsity patterns using the Gerschgorin Circle Theorem is $O(n)$ with selection algorithms [40]. Repeating this for all variables costs $O(n^2)$. For the algorithms based on Frobenius norm and 1-norm of the matrix at iteration i, $i(n-i)$ flops are required to calculate the effect of adding any variable k to the current set. Finding the maximum over this list costs $O(n)$. Hence, the costs of constructing and repeating for these algorithms are $O(ns^2/2 - s^3/6)$ and $O(n^2s^2/2 - ns^3/6)$, respectively. Finally, the additional cost of finding the dominant eigenvalue for all sparsity patterns and selecting the maximum is $O(ns^2)$ and $O(n)$, respectively. The computational complexity of our proposed algorithms is summarized in Table 3.1.

Algorithm	GD	CCW	FCW
Computational Complexity	$O(ns^2 + n^2)$	$O(n^2s^2)$	$O(n^2s^2)$

Table 3.1: Computational Complexity of the Proposed Greedy Algorithms

3.3.5 Shrinking the Search Space

The computational cost of our algorithms is relatively high and most of the time, good sparsity patterns are constructed from only a few variables. Hence, to save computational effort, we modify the repetitive step of the algorithm. Instead of performing a full search starting from each variable, we construct patterns from t variables where $t < n$. These t variables are chosen so that they have the largest column norms. This method successfully finds strong candidates in practice. We also used modified versions of our algorithms in our tests.

3.4 Enforcing Necessary Conditions

Algorithms presented in Section 3.3 are based on greedy heuristics that compute the largest upper bounds for the dominant eigenvalue derived in section 3.2. After termination of the algorithms, final sparsity patterns may not satisfy necessary optimality conditions like stationarity or CW-Optimality. This means there could be room for further improvements on the final output of the algorithms based on these optimality conditions.

In this section we focus on the following problem;

$$\begin{aligned} &\max f(x) \\ &x \in \mathcal{X} \end{aligned} \tag{3.13}$$

Where $f(x)$ is a differentiable function and $\mathcal{X}$ is a nonempty compact set . For the SPCA problem, we have $f(x) = x^T Ax$ and $\mathcal{X} = \{x \in \mathbb{R}^n \,|\, ||x||_2 = 1,\, ||x||_0 \leq s\}$. For the model given in (3.13), we consider two necessary optimality conditions; Stationarity and Coordinate-wise (CW) optimality.

3.4.1 Stationarity

Let x be local maximum of the problem (3.13). Then it must satisfy the following inequality [41];

$$\nabla f(x)^T(z-x) \leq 0 \qquad \forall z \in \mathcal{X}. \tag{3.14}$$

This general first-order condition is referred to as the *stationarity condition*. For this condition to be necessary, we must assume convexity for the function f or the set $\mathcal{X}$. In our setting f is convex because it is a quadratic function with matrix A which is positive semi-definite as it is a variance/covariance matrix. Generalized Power method and Conditional Gradient algorithms suggested in [16, 17] are based on computing points that satisfy this condition on different formulations of SPCA. In addition to the original formulation, they also discuss ℓ_0 norm penalized, ℓ_1 norm penalized, and constrained versions of the problem.

3.4.2 CW-Optimality

Coordinate-wise optimality condition is a local optimality condition over a subset of variables. In sparsity constrained problems the neighborhood of a point where the optimality condition is checked corresponds to the set of points that differ in their support. Let x be coordinate-wise optimal point for problem (3.13) then it must satisfy the following;

$$f(x) \geq f(z) \qquad \forall z \in D_k(x),$$

where $D_k(x) = \{z \,|\, |I(z) \setminus I(x)| \leq k,\, z \in \mathcal{X}\}$ is the neighborhood set and $I(x) = \{i \,|\, x_i \neq 0\}$ is the indicator function. CW-Optimality condition gives vectors where the cardinality of support differs in at most k elements compared to the support of x, and which do not have a better objective value than x. Beck and Vaisbourd [20] show that the CW-Optimality condition is more restricting than the stationarity condition for problem (3.13), and developed algorithms based on finding points that satisfy CW-Optimality conditions.

3.4.3 Enhancing the Solutions via Necessary Conditions

Based on the necessary optimality conditions given in the previous subsections, we can enforce stationarity or CW-Optimality conditions on sparsity patterns generated by algorithms proposed in section 3.3 to improve the quality of the solutions.

Since we focus on the ℓ_0 constrained problem, we can use the following iteration suggested in [17] until convergence to enforce the stationarity condition;

$$x^{j+1} = \frac{T_s(Ax^j)}{||T_s(Ax^j)||}, \quad \text{where} \tag{3.15}$$

$$T_s(a) = \arg\min_x \{||x-a||_2^2 \,;\, ||x||_0 \leq s, x \in \mathbb{R}^n\} \tag{3.16}$$

$$= \{s \text{ largest elements of } a\}. \tag{3.17}$$

This iteration may also be recognized as the Power method for calculating the dominant eigenvalue of the matrix A with additional sparsity constraint explicitly enforced at each iteration [38]. Convergence and recovery of sparse components with this algorithm are analyzed in [42]. Furthermore, the sparse eigenvector x can be constructed by the zero-padding operation.

In order to enforce the CW-Optimality condition to the solution x, we need to check all possible support candidates that differ from that of x in at most two nonzero indices. Let us define $I^C(x) = \{i \,|\, x_i = 0\}$ as the index set of variables that are equal to zero. Let P and x_P, λ_P be the pattern generated by any of the algorithms and corresponding sparse eigenpair. Here, one could also choose to follow the second part of the PCW algorithm suggested in [20] to enforce this condition. However, enforcing the CW-Optimality condition is computationally much more expensive than checking the stationarity condition. Hence, in our algorithms, we only use the stationarity condition which is easier to verify.

3.5 Multiple Components and Deflation

Generally, more than one principal component will be necessary to capture most of the variance in the data. In PCA, the best k components correspond to finding largest k dominant eigenvectors of the covariance matrix A [43]. As a result, loadings are orthogonal and components are uncorrelated. In SPCA, to compute multiple components, we iteratively deflate the data matrix using the well-known orthogonal subspace projection method:

$$B^{(r)} = B(I - xx^T) \tag{3.18}$$

$$A^{(r)} = (I - xx^T)A(I - xx^T), \tag{3.19}$$

where x is the vector that solves the SPCA problem, $B \in \mathbb{R}^{m \times n}$ is the data matrix and $A \in \mathbb{R}^{n \times n}$ covariance matrix. We note that there exist more involved methods of deflation discussed in [44].

The loadings constructed with this procedure do not necessarily result in orthogonal vectors. In addition, components may not be uncorrelated either. Therefore, summing up the sparse eigenvalues will overestimate variance jointly explained by all components. Hence, adjusted variance introduced in [13] should be used to estimate jointly explained total variance. Let $P \in \mathbb{R}^{n \times k}$ denote k loadings. Adjusted variance explained by the components is given as

$$\begin{aligned} C &= BP \\ C &= QR \\ \widehat{Var(C)} &= \textstyle\sum_{i=1}^{k} R_{ii}^2, \end{aligned} \tag{3.20}$$

where $C = QR$ is the QR decomposition of the matrix C, $Q \in \mathbb{R}^{n \times k}$ has orthogonal columns and $R \in \mathbb{R}^{k \times k}$ is upper triangular. Moreover, if the data matrix is not available we have

$$\begin{aligned} D &= P^T AP \\ D &= LL^T \\ \widehat{Var(D)} &= \textstyle\sum_{i=1}^{k} L_{ii}^2, \end{aligned} \tag{3.21}$$

where $D = LL^T$ is the Cholesky decomposition of the matrix D and $L \in \mathbb{R}^{k \times k}$ is a lower triangular matrix.

Chapter 4

PCA Sparsified

In this chapter, we develop a new algorithm that uses a non-convex log-sum penalty approach in the space of positive semidefinite matrices. Our algorithm leverages the DSPCA [21] approach, which is a convex semidefinite relaxation approach to SPCA, which has remarkable numerical results and reweighted ℓ_1 minimization [45, 46] which is known to enhance the sparsity of the solutions over classical ℓ_1 norm penalty approach.

4.1 Notation

We use $\mathbf{S}^n$ for the set of symmetric matrices of size n and $\mathbf{S}^n_+$ for symmetric positive definite matrices of size n. $\langle \cdot, \cdot \rangle$ denotes the inner product and $X \circ Y$ denotes Hadamard, or element-wise, product of matrices X and Y. We denote the element-wise power of matrix X by $X^{\circ n}$ for $n \in \mathbb{R}$. The ℓ_1 norm of a matrix is defined as $||X||_1 = \sum_{i=1}^n \sum_{j=1}^n |x_{ij}|$. We let $\mathbf{1}$ represent a vector with all components equal to one. The indicator function of a set $\mathcal{X}$ is defined as

$$\mathcal{I}_{\mathcal{X}}(X) = \begin{cases} 0 & \text{if } X \in \mathcal{X} \\ +\infty & \text{otherwise.} \end{cases} \tag{4.1}$$

The proximal operator of a function f is defined as

$$\text{prox}_f(x) = \arg\min_y f(y) + \tfrac{1}{2}||y - x||_2^2. \tag{4.2}$$

The vec operator constructs a column vector from a given matrix $A \in \mathbb{R}^{m\times n}$:

$$\text{vec}(A) = \Big[a_{11}, a_{12}, \ldots, a_{1n}, a_{21}, \ldots, a_{mn}\Big]^T. \tag{4.3}$$

Conversely mat operator constructs a symmetric matrix from a given vector $a \in \mathbb{R}^{n^2}$

$$\text{mat}(a) = \begin{pmatrix} a_1 & a_{n+1} & \ldots & a_{n^2-n+1} \\ a_2 & a_{n+2} & \ldots & a_{n^2-n+2} \\ \vdots & \vdots & \ddots & \vdots \\ a_n & a_{2n} & \ldots & a_{n^2} \end{pmatrix}. \tag{4.4}$$

Column or row orientation of construction does not change the matrix since we will only apply mat() operator to a vector constructed by applying vec() operator on a symmetric matrix.

4.2 New Perspective for SPCA

Most of the current approaches solve the sparsity constrained or the sparsity penalized variance maximization problems as posed in (1.2). This is accomplished either by optimizing a local approximation or by solving convex relaxations of the original problem. In this chapter, we focus on the minimization of the cardinality of the loadings subject to a variance constraint ("inverted SPCA" approach) which is formulated as follows

$$\begin{aligned} \min_x \quad & ||x||_0 \\ \text{st.} \quad & ||x||_2 = 1 \\ & x^T A x \geq b. \end{aligned} \tag{4.5}$$

Our motivation for this inverted approach comes from SPCA practice. There is no optimal way of selecting the sparsity level for the original SPCA problem.

In practice, the problem is solved for different sparsity levels, and the trade-off between sparsity and variance is analyzed. Then, an appropriate sparsity level is chosen. In a similar fashion, with our model, the problem can be solved for several levels of variance and appropriate variance level and sparsity can be chosen accordingly. In section 4.3 we derive a more tractable approach to solve (4.5) and illustrate with numerical experiments (in section 5) that our approach squeezes out more variance or equivalently more sparsity than other methods by leveraging reweighted optimization.

4.3 Semidefinite Relaxation

After lifting the problem into the symmetric matrix space we relax the cardinality penalty term and replace it with an ℓ_1 norm, which is a common way to get a convex approximation in the literature [47, 48, 49]. This relaxation is also referred to as the operation of replacing the function with its convex envelope. Indeed, the ℓ_1 norm is the convex envelope of the ℓ_0 norm on the unit ball. Moreover, in order to handle the non-convex quadratic constraint, we use the semidefinite lifting method as in [50, 21, 51] to get

$$\begin{aligned} \min_{X} \quad & ||X||_1 \\ \text{st.} \quad & \text{tr}(AX) \geq b \\ & \text{tr}(X) = 1 \\ & X \succeq 0 \\ & \textbf{Rank}(X) = 1. \end{aligned} \tag{4.6}$$

Omission of the rank constraint leads to a convex optimization problem in matrix variable $X \in \mathbf{S}^n$. Moreover, we assume that $\max_i \{A_{ii}\} = d_{max} < b \leq \lambda_1(A)$ so that the variance constraint is always tight and the problem is feasible. If $b > \lambda_1(A)$, the problem is infeasible as the maximum of $\text{tr}(AX)$ under the restrictions $X \succeq 0, \text{tr}(X) = 1$ is equal to $\lambda_1(A)$. Similarly, if $b \leq d_{max}$ then the problem is not interesting. There could be infinitely many solutions in the optimal set among which one has $X = e_j e_j^T$ where $d_{max} = d_j$ is one.

4.3.1 Reweighted Optimization

In this study, we consider a variant of the optimization problem (4.6), where instead of using ℓ_1 norm of the matrix to induce sparsity and thus obtaining a model equivalent to DSPCA [21], we use a concave log-sum penalty function. The same approach was used in [45] for computing sparse solutions to a linear system and in [46] for finding low-rank solutions and sparsity in singular values, subject to affine constraints. Hence, we consider the following problem

$$\begin{aligned} \min_{X} \quad & \tilde{g}(X) = \textstyle\sum_{i=1}^{n} \sum_{j=1}^{n} \log(\epsilon + |x_{ij}|) \\ \text{st.} \quad & \operatorname{tr}(AX) = b \\ & \operatorname{tr}(X) = 1 \\ & X \succeq 0, \end{aligned} \tag{4.7}$$

where $\epsilon > 0$ is a stability parameter to make log-sum penalty function differentiable everywhere. Since the problem (4.7) consists of minimization of a concave function over a convex set, any local minimum has to satisfy the First-Order Condition [52]. However, since the absolute value is not differentiable everywhere, consider an equivalent formulation to (4.7) by introducing an auxiliary variable U

$$\begin{aligned} \min_{X,U} \quad & g(U) = \textstyle\sum_{i=1}^{n} \sum_{j=1}^{n} \log(\epsilon + u_{ij}) \\ \text{st.} \quad & \operatorname{tr}(AX) = b \\ & \operatorname{tr}(X) = 1 \\ & X \succeq 0. \\ & |x_{ij}| \leq u_{ij} \quad \forall i, j = 1, .., n. \end{aligned} \tag{4.8}$$

For conciseness, let C denote the optimization domain. The first-order condition states that if $Y = (X, U)$ is a local minimum of a concave function g over a compact set C it must satisfy

$$\langle Z - Y, \nabla g(Y) \rangle \geq 0 \quad \forall\, Z \in C. \tag{4.9}$$

This suggests the following algorithm (PCA Sparsified) for solving (4.7): start with some feasible point $(X^{(0)}, U^{(0)})$, and iteratively minimize a linearized version

of g

$$\begin{aligned}(X^{(t+1)}, U^{(t+1)}) = &\quad \arg\min_{X,U} \left\{ \langle \nabla g(U^{(t)}), U \rangle : (X, U) \in C \right\} \\ &[\nabla g(U^{(t)})]_{ij} = \frac{1}{u_{ij}^{(t)} + \epsilon},\end{aligned} \tag{4.10}$$

which is equivalent to the following problem

$$X^{(t+1)} = \quad \arg\min_{X} \left\{ \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{|x_{ij}|}{|x_{ij}^{(t)}| + \epsilon} : \operatorname{tr}(AX) = b,\ \operatorname{tr}(X) = 1,\ X \succeq 0 \right\}. \tag{4.11}$$

The additional variable can be extracted as $u_{ij}^{(t+1)} = |x_{ij}^{(t+1)}|$. Defining a weighting matrix W as $w_{ij} = \frac{1}{|x_{ij}^{(t)}| + \epsilon}$, as a placeholder for the gradient of the surrogate function, we can recognize the minimization problem at each iteration (4.11) as a convex optimization problem in the form of (4.12)

Algorithm 4: PCA Sparsified for solving (4.7)

Input: $\epsilon > 0, T \in \mathbb{Z}_+$
Set $W^{(0)} = \mathbf{1}\mathbf{1}^T, t = 0, \lambda_P = 0, v_P = \emptyset, P = \emptyset$
while $W^{(t)}$ not converged **and** $t \leq T$ **do**
  Solve (4.12) using CGAL or ADMM for the current weights $W^{(t)}$
  Extract $\lambda_1(X^{(t)}), v_1(X^{(t)})$, the dominant eigen-pair of $X^{(t)}$
  Compute pattern S, largest s elements of $v_1(X^{(t)})$ in magnitude
  Compute eigenpair $\lambda_1(A_S), v_1(A_S)$, where A_S is the submatrix corresponding to variable set S
  if $\lambda_1(A_S) > \lambda_P$ **then**
    Update $\lambda_P := \lambda_1(A_S)$
    Update $v_P := v_1(A_S)$
    Update $P := S$
  end if
  Update $w_{ij}^{(t+1)} := (\epsilon + |x_{ij}^{(t)}|)^{-1} \quad \forall\, i, j$
  Update $t := t + 1$
end while
return λ_P, v_P, P

$$\begin{aligned}\min_{X} \quad & ||W \circ X||_1 \\ \text{st.} \quad & \operatorname{tr}(AX) = b \\ & \operatorname{tr}(X) = 1 \\ & X \succeq 0.\end{aligned} \tag{4.12}$$

We see that at each iteration our iterative algorithm solves a weighted version of the problem (4.6). At iteration k, weights are set as inversely proportional to the magnitudes of the solution of the problem at iteration $(k-1)$. As suggested in [53], we initialize the algorithm with weights all equal to 1, i.e., unweighted ℓ_1 norm. We note that the scale of weights is not important, i.e., we can multiply all the weights by a positive constant, which will not change the result of the algorithm. This is the reason why we use a variance constraint in our approach instead of adding a penalty term. Otherwise, either the scale of the weights or the penalty term related to variance would have to be adjusted at each iteration after computing new weights.

This iterative algorithm is an example of Majorization-Minimization (MM) [54] algorithms.[1] MM algorithms achieve minimization of a given objective function by iteratively minimizing a surrogate function that majorizes it. In our approach, we use the linearization of the log-sum penalty function as the surrogate since the tangent of a concave function always remains above (or glued to) the function.

Figure 4.1 illustrates the strength of the log-sum penalty function in terms of its ability to approximate the ℓ_0 norm better than the ℓ_1 norm. Thus, it is expected that the log-sum penalty would induce more sparsity. However, while changing the log-sum penalty for the ℓ_1 norm, the power of convex optimization is lost. Consequently, we resort to local minimization techniques for minimizing the log-sum penalty function.

4.3.2 Connection to DSPCA

Our proposed model is closely related to the previously suggested DSPCA model in [21]. In this section, we show the equivalence of a subproblem solved by PCA Sparsified and DSPCA. Consider the first step of the algorithm where $W_0 = \mathbf{1}\mathbf{1}^T$.

[1] This algorithm can also be interpreted as CG Algorithm applied to surrogate function log-sum penalty with a unit step size. However, in order to avoid confusion with CG in CGAL algorithm for the subproblem, we present PCA Sparsified as an MM algorithm.

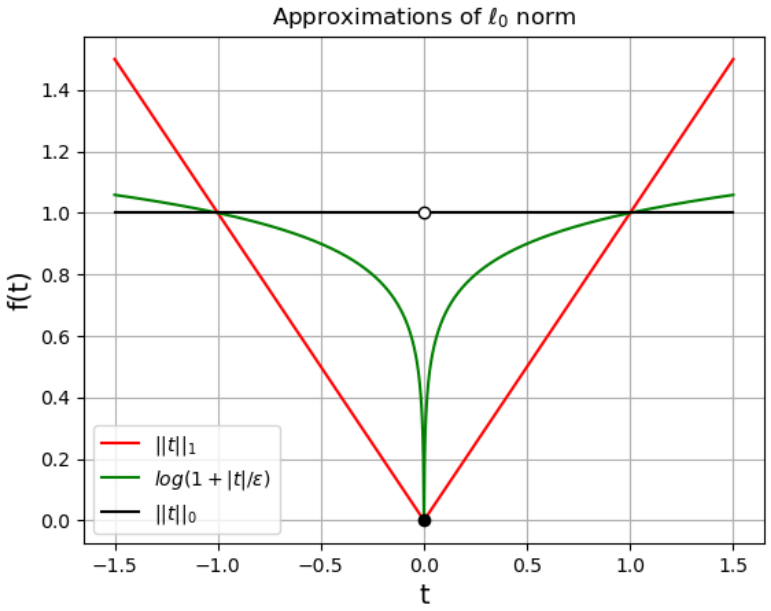


Figure 4.1: Around the origin log-sum function penalizes t with large values, and for large values of t, penalty grows much slower than convex ℓ_1 relaxation, which is why log-sum is a better candidate for approximating ℓ_0 norm

We start by dualizing the affine variance constraint:

$$\begin{aligned} \min_X \quad & ||X||_1 - \xi \operatorname{tr}(AX) \\ \text{st.} \quad & \operatorname{tr}(X) = 1 \\ & X \succeq 0. \end{aligned} \tag{4.13}$$

By our assumptions on b, we know that $\xi > 0$. Scaling by the dual variable ξ and rewriting the problem equivalently using $\min f = -\max -f$ we obtain

$$\begin{aligned} \max_X \quad & \operatorname{tr}(AX) - \xi^{-1}||X||_1 \\ \text{st.} \quad & \operatorname{tr}(X) = 1 \\ & X \succeq 0 \end{aligned} \tag{4.14}$$

Now the connection between PCA Sparsified and DSPCA becomes clear. Model (4.14) is the "Robust Maximum Eigenvalue problem" [21] with $\rho = \xi^{-1}$. Thus, the first step of PCA Sparsified for some level of variance constraint is equivalent to DSPCA for some level of ℓ_1 norm penalty or ℓ_1 norm constraint. Moreover,

equivalent formulations can be written in terms of one another using optimal Lagrange multipliers. This simple observation shows that convex relaxations of SPCA formulations are equivalent: sparsity constrained/penalized variance maximization and variance constrained sparsity minimization. On a final note, subproblems solved in the subsequent iterations of PCA Sparsified would use an arbitrary weighting matrix W. Such a subproblem would be equivalent to a weighted ℓ_1 penalty version of DSPCA.

4.4 Algorithms for the Subproblem of PCA Sparsified

For small problem instances, the optimization problem (4.12) can be solved using interior point methods. However, due to their $O(n^6)$ complexity per iteration and large memory requirements, interior point methods become quickly impractical for large problem instances. To overcome these difficulties we propose to deploy two different first-order algorithms: Alternating Directions Method of Multipliers (ADMM) [55] and Conditional Gradient Augmented Lagrangian (CGAL) method [56]. We discuss these methods next.

4.4.1 ADMM

ADMM operates on the optimization problem by dividing the global optimization effort into small subproblems which have simple, either computationally cheap or closed-form, solutions. We obtain these closed-form solutions for the ADMM steps by utilizing proximal operators [57].

After introducing extra variables $X = [X_1, X_2, X_3]$, we can rewrite the problem

(4.12) as

$$\begin{array}{ll} \min\limits_{X} & ||W \circ X_1||_1 \\ \text{st.} & X_1 = X_2 \\ & \text{tr}(AX_2) = b \\ & \text{tr}(X_2) = 1 \\ & X_2 = X_3 \\ & X_3 \in \mathbf{S}_+^n. \end{array} \tag{4.15}$$

Then, we construct the corresponding augmented Lagrangian [58]

$$\begin{aligned} \mathcal{L}_\rho(X, y, U) = &\|W \circ X_1\|_1 + \mathcal{I}_{\mathbf{S}_+^n}(X_3) + y_1(\text{tr}(AX_2) - b) + y_2(\text{tr}(X_2) - 1) \\ &+\langle U_1, X_1 - X_2\rangle + \langle U_2, X_3 - X_2\rangle + (\rho/2)(\text{tr}(AX_2) - b)^2 + (\rho/2)(\text{tr}(X_2) - 1)^2 \\ &+(\rho/2)\|X_1 - X_2\|_F^2 + (\rho/2)\|X_3 - X_2\|_F^2, \end{aligned} \tag{4.16}$$

where $U = [U_1, U_2]$ and $y = [y_1, y_2]$ are dual variables and ρ is the augmented Lagrangian parameter, and the convexity constraint on X_3 was replaced by the indicator function of the set.

ADMM at iteration number k is comprised of the following updates:

1. $X_1^{(k+1)} = \arg\min\limits_{X_1} \mathcal{L}_\rho(X_1, X_2^{(k)}, X_3^{(k)}, y^{(k)}, U^{(k)})$
2. $X_2^{(k+1)} = \arg\min\limits_{X_2} \mathcal{L}_\rho(X_1^{(k+1)}, X_2, X_3^{(k)}, y^{(k)}, U^{(k)})$
3. $X_3^{(k+1)} = \arg\min\limits_{X_3} \mathcal{L}_\rho(X_1^{(k+1)}, X_2^{(k+1)}, X_3, y^{(k)}, U^{(k)})$
4. $y^{(k+1)} = \begin{bmatrix} y_1^{(k+1)} \\ y_2^{(k+1)} \end{bmatrix} = \begin{bmatrix} y_1^{(k)} \\ y_2^{(k)} \end{bmatrix} + \rho \begin{bmatrix} \text{tr}(AX_2^{(k+1)}) - b \\ \text{tr}(X_2^{(k+1)}) - 1 \end{bmatrix}$
5. $U^{(k+1)} = \begin{bmatrix} U_1^{(k+1)} \\ U_2^{(k+1)} \end{bmatrix} = \begin{bmatrix} U_1^{(k)} \\ U_2^{(k)} \end{bmatrix} + \rho \begin{bmatrix} X_1^{(k+1)} - X_2^{(k+1)} \\ X_3^{(k+1)} - X_2^{(k+1)} \end{bmatrix}.$

Steps 1, 2, and 3 minimize the augmented Lagrangian function over X_i's sequentially. Steps 4 and 5 perform updates of the dual variables according to the infeasibility gap. We now examine these steps in detail.

X-Updates. The X update is split into three parts: X_1 does minimization over the ℓ_1 norm which promotes sparsity, X_3 ensures positive semi-definiteness of the variable, and X_2 both handles the affine constraints and ensures consensus between X_1 and X_3.

X_1 update: X_1 update can be written in terms of proximal operator of ℓ_1 norm after completing the square in (4.16) with respect to X_1 related terms. It has a simple closed-form solution:

$$\begin{aligned} X_1^{(k+1)} &= \arg\min_{X_1} ||W \circ X_1||_1 + (\rho/2)||X_1 - X_2^{(k)} + \rho^{-1}U_1^{(k)}||_F^2 \\ &= \text{prox}_{||\rho^{-1}W\cdot||_1}(X_2^{(k)} - \rho^{-1}U_1^{(k)}) = S_{\rho^{-1}W}(X_2^{(k)} - \rho^{-1}U_1^{(k)}), \end{aligned} \tag{4.17}$$

where S is a soft thresholding operator which is defined as

$$[S_{\rho^{-1}W}(Y)]_{ij} = \begin{cases} 0 & if\ |[Y]_{ij}| \leq W_{ij}/\rho \\ \text{sign}([Y]_{ij})(|[Y]_{ij}| - W_{ij}/\rho) & \text{otherwise.} \end{cases} \tag{4.18}$$

X_2 update: X_2 update is obtained from a quadratic optimization problem which has a closed-form solution:

$$\begin{aligned} X_2^{(k+1)} &= \arg\min_{X_2} \tfrac{\rho}{2}||X_1^{(k+1)} - X_2 + \rho^{-1}U_1^{(k)}||_F^2 + \tfrac{\rho}{2}||X_3^{(k)} - X_2 + \rho^{-1}U_2^{(k)}||_F^2 \\ &+ \tfrac{\rho}{2}(tr(AX_2) - b + \rho^{-1}y_1^{(k)})^2 + \tfrac{\rho}{2}(tr(X_2) - 1 + \rho^{-1}y_2^{(k)})^2. \end{aligned} \tag{4.19}$$

In order to view the above problem as a quadratic optimization problem we map the symmetric matrix variable X_2 to a vector, $x_2 = \text{vec}(X_2)$. Then, the X_2 update subproblem becomes:

$$\begin{aligned} x_2^{(k+1)} &= \arg\min_{x_2} \tfrac{\rho}{2}||x_1^{(k+1)} - x_2 + \rho^{-1}u_1^{(k)}||_2^2 + \tfrac{\rho}{2}||x_3^{(k)} - x_2 + \rho^{-1}u_2^{(k)}||_2^2 \\ &+ \tfrac{\rho}{2}(a^T x_2 - b + \rho^{-1}y_1^{(k)})^2 + \tfrac{\rho}{2}(v^T x_2 - 1 + \rho^{-1}y_2^{(k)})^2, \end{aligned} \tag{4.20}$$

where we also mapped other variables and matrices to a vector, i.e., we have $x_1^{k+1} = \text{vec}(X_1^{(k+1)})$, $x_3^{(k)} = \text{vec}(X_3^{(k)})$, $u_1^{(k)} = \text{vec}(U_1^{(k)})$, $u_2^{(k)} = \text{vec}(U_2^{(k)})$, $a = \text{vec}(A)$, $v = \text{vec}(I_n)$. Then, the solution to the quadratic optimization problem

is given by:

$$(2I_{n^2} + aa^T + vv^T)x_2^{(k+1)} = \begin{aligned} & x_1^{(k+1)} + \rho^{-1}u_1^{(k)} + x_3^{(k)} + \rho^{-1}u_2^{(k)} + ba - \rho^{-1}y_1^{(k)}a + \\ & v - \rho^{-1}y_2^{(k)}v. \end{aligned} \tag{4.21}$$

Hence, the X_2 update is equivalent to solving a linear system of size n^2. The term aa^T almost guarantees that the matrix $2I_{n^2} + aa^T + vv^T$ will always be dense. Thus, the computational cost of this operation will be $O((n^2)^3)$. Fortunately, this matrix operation can be performed using a number of operations much smaller than $O(n^6)$ using the matrix inversion lemma [59] since the identity matrix is invertible and the terms aa^T and vv^T are symmetric rank one updates:

$$\begin{aligned} & (2I_{n^2} + aa^T + vv^T)^{-1} = (1/2)I_{n^2} - (1/4)V(I_2 + (1/2)V^TV)^{-1}V^T \\ & f = x_1^{(k+1)} + \rho^{-1}u_1^{(k)} + x_3^{(k)} + \rho^{-1}u_2^{(k)} + ba - \rho^{-1}y_1^{(k)}a \\ & x_2^{(k+1)} = (2I_{n^2} + aa^T + vv^T)^{-1}f, \end{aligned} \tag{4.22}$$

where $V = [a \ v]$. Mapping the vector variable x_2 back to matrix space, $X_2 = \text{mat}(x_2)$ completes the X_2 update. We remark that the matrix-vector multiplication in (4.22) does not result in $O((n^2)^2)$ computational cost as the operation can be carried out by multiplying f with V, which costs only $O(n^2)$.

X_3 update: Similar to that of X_1, X_3 update can also be written in terms of proximal operators

$$\begin{aligned} X_3^{(k+1)} &= \arg\min_{X_3} \mathcal{I}_{\mathbf{S}_+^n}(X_3) + (\rho/2)||X_3 - X_2^{(k+1)} + \rho^{-1}U_2^{(k)}||_F^2 \\ &= \text{prox}_{\mathcal{I}_{\mathbf{S}_+^n}}(X_2^{(k+1)} - \rho^{-1}U_2^{(k)}) = \mathcal{P}_{\mathbf{S}_+^n}(X_2^{(k+1)} - \rho^{-1}U_2^{(k)}). \end{aligned} \tag{4.23}$$

It is well known that the proximal operator of the indicator function of a convex set is the projection operator. Calculation of the projection requires a full eigendecomposition, and taking the non-negative part of the eigenvalues. This

operation can be justified as follows;

$$
\begin{aligned}
Z &= X_2^{(k+1)} - \rho^{-1} U_2^{(k)} = QDQ^T \\
X_3^{(k+1)} &= \underset{X_3 \in \mathbf{S}_+^n}{\arg\min}\ (1/2)||X_3 - Z||_F^2 = (1/2)||X_3 - QDQ^T||_F^2 \\
&= \underset{X_3 \in \mathbf{S}_+^n}{\arg\min}\ (1/2)||Q^T X_3 Q - D||_F^2 \\
&= \underset{\hat{X}_3 \in \mathbf{S}_+^n}{\arg\min}\ (1/2)||\hat{X}_3 - D||_F^2 = \textstyle\sum_{i=1}^n (\hat{X}_{3_{ii}} - D_{ii})^2 \\
\hat{X}_{3_{ii}} &= max(D_{ii}, 0) \quad \forall\, i = 1, \ldots, n \\
X_3^{(k+1)} &= Q\hat{X}_{3_{ii}} Q^T
\end{aligned}
\tag{4.24}
$$

In the first line, we define a placeholder matrix Z for the complicated expression with a primal variable and a scaled dual variable. Then we use the eigendecomposition of Z to simplify the objective in the following lines because orthogonal transforms do not change the Frobenius norm, which is sometimes referred to as unitarily invariant [39]. In fact, what we apply here can also be viewed as a similarity transform which also preserves eigenvalues and hence preserves the Frobenius norm. Then, the application of the similarity transform to the X_3 variable allows us to compute the solution to the new problem easily. Transforming the variable back gives the solution to the original problem.

Convergence. Although the problem (4.15) is convex, the classical theory of ADMM guarantees convergence to a global optimum for the two-block case. It was shown in [60] that it is not necessarily the case for three or more blocks. However, they also provide a sufficient condition for convergence for the multi-block ADMM. Our model satisfies this sufficient condition, and therefore ADMM is guaranteed to converge to a global optimum for our model as well. In Appendix A we show that the ADMM subproblem for PCA Sparsified indeed satisfies the sufficient condition. Furthermore, for better convergence in practice, we used an adaptive augmented Lagrangian penalty scheme described in [55] which keeps primal and dual residual norms within a constant factor of each other. A stopping criterion based on primal and dual residual norms being below certain thresholds or a maximum number of ADMM iterations can be specified for the termination of the algorithm. Derivation of primal and dual residuals can be found in Appendix

B.

4.4.2 CGAL

The power of CGAL stems from computationally inexpensive conditional gradient steps, so-called linear minimization oracles, on the primal variable. In order to obtain a computationally efficient CGAL scheme, we need to pose the problem on a compact (nonempty) set on which the linear minimization oracle will be called. Hence, we cast problem (4.12) as follows:

$$\begin{aligned} \min_{X} \quad & ||W \circ X||_1 \\ \text{st.} \quad & \text{tr}(AX) = b \\ & X \in \mathcal{X}, \end{aligned} \tag{4.25}$$

where $\mathcal{X} = \{X \mid \text{tr}(X) = 1, X \succeq 0\}$ is the compact domain of optimization variable X. This choice of compact optimization domain requires an eigenvector computation as a linear minimization oracle in the conditional gradient steps. This can be accomplished efficiently using Krylov subspace methods as well as the simple power method.

Similar to ADMM, CGAL also operates on the augmented Lagrangian [58]. For problem (4.25) it corresponds to the following

$$\mathcal{L}_\rho(X, y) = \|W \circ X\|_1 + y(\text{tr}(AX) - b) + (\rho/2)(\text{tr}(AX) - b)^2. \tag{4.26}$$

CGAL is specifically designed to be able to provably solve non-smooth problems. Unlike CGAL, other CG-type methods cannot provably solve non-smooth problems, a counter example is given by Nesterov [61]. CGAL handles the non-smooth objective by using a smoothing technique described in [62] commonly known as "Nesterov Smoothing" and gradually decreasing the smoothing level. Specifically, at each iteration we minimize a smooth approximation to (4.26). Since in our problem we use a non-smooth ℓ_1 norm penalty to induce sparsity,

we need to smooth it to be able to use CGAL.

We consider the following smooth function that approximates $||W \circ X||_1$;

$$||W \circ X||_{H_\beta} = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} H_\beta(x_{ij}), \tag{4.27}$$

where $H_\beta(x)$ is the Huber loss function defined as

$$H_\beta(x) = \begin{cases} \dfrac{x^2}{2\beta} & if -\beta \leq X \leq \beta \\ |x| - \frac{\beta}{2} & \text{otherwise.} \end{cases} \tag{4.28}$$

Moreover, the corresponding gradient can be computed as follows

$$[\nabla ||X||_{H_\beta}]_{ij} = \begin{cases} \dfrac{w_{ij} x_{ij}}{\beta} & if -\beta \leq x_{ij} \leq \beta \\ \text{sign}(x_{ij}) w_{ij} & \text{otherwise.} \end{cases} \tag{4.29}$$

More details and derivations of the smooth function can be found in Appendix C.

Replacing the non-smooth function with the smooth function the augmented Lagrangian becomes

$$\mathcal{L}_{\rho,\beta}(X, y) = ||W \circ X||_{H_\beta} + y(\text{tr}(AX) - b) + (\rho/2)(\text{tr}(AX) - b)^2. \tag{4.30}$$

The gradient of the augmented Lagrangian with respect to the primal variable X can be computed as

$$\nabla_X \mathcal{L}_{\rho,\beta}(X, y) = \nabla ||W \circ X||_{H_\beta} + yA + \rho(\text{tr}(AX) - b)A. \tag{4.31}$$

CGAL at iteration k consists of the following updates:

1. $\eta_k = 2/(k+1)$, $\rho_k = \rho_0 \sqrt{k+1}$, $\beta_k = \beta_0 / \sqrt{k+1}$
2. $S^{(k)} = \arg\min_{X \in \mathcal{X}} \langle \nabla_X \mathcal{L}_{\rho_k, \beta_k}(X^{(k)}, y^{(k)}), X \rangle$

3. $X^{(k+1)} = (1-\eta_k)X^{(k)} + \eta_k S_{(k)}$

4. $y^{(k+1)} = y^{(k)} + \gamma_{k+1}(\mathrm{tr}(AX^{(k+1)}) - b)$,

where ρ_0 and β_0 are initial input parameters. The first step adjusts the step size, the augmented Lagrangian parameter, and the smoothing parameter. The second and third steps are the conditional gradient steps on the primal variable X. The last step updates the dual variable y according to the infeasibility gap. Now, we discuss these steps in more detail.

X-Update. The X update corresponds to a linear minimization oracle followed by taking a small step in the direction of the minimizer. Let $G^{(k)}$ denote the partial derivative matrix of the augmented Lagrangian with respect to X at step k. Then the linear minimization becomes

$$\begin{aligned} S^{(k)} &= \arg\min \left\{ \langle G^{(k)}, X \rangle : \mathrm{tr}(X) = 1, X \succeq 0 \right\} \\ &= v_n(G^{(k)}) v_n(G^{(k)})^T, \end{aligned} \tag{4.32}$$

where $v_n(G^{(k)})$ is the eigenvector corresponding to the smallest eigenvalue of $G^{(k)}$. Then, we take a small step in the steepest descent direction

$$X^{(k+1)} = (1-\eta_k)X^{(k)} + \eta_k S^{(k)}, \tag{4.33}$$

where $\eta_k = 2/(k+1)$ is a general step size rule for CG-type methods.

y-Update. Dual update on y is performed using gradient ascent which amounts to a correction by infeasibility gap in the variance constraint:

$$y^{(k+1)} = y^{(k)} + \gamma_{k+1}(\mathrm{tr}(AX^{(k+1)}) - b). \tag{4.34}$$

We use the constant step size rule of CGAL. Hence, we choose the largest $\gamma_{k+1} \geq 0$ that satisfies

$$\begin{aligned} \gamma_{k+1} &\leq \rho_0 \\ \gamma_{k+1}(\mathrm{tr}(AX^{(k+1)}) - b)^2 &\leq \tfrac{1}{2}\eta_k^2 L_{k+1} D_{\mathcal{X}}^2 \\ y^{(k+1)} &\leq D_{y^{(k+1)}} \end{aligned} \tag{4.35}$$

where $D_{\mathcal{X}} = \max_{X_1, X_2 \in \mathcal{X}} ||X_1 - X_2||_F = \sqrt{2}$ is the diameter of the compact optimization domain, $D_{y^{(k+1)}} = D_y \in \mathbb{R}_+$ is the dual domain diameter at step $k+1$, and

$$L_{k+1} = \rho_{k+1}||A||_F^2 + \beta_{k+1}^{-1}||W||^2$$

is the Lipschitz constant of the augmented Lagrangian at step $k+1$. While the first two constraints are explicit on dual step size, the third one implicitly constrains the dual step size by ensuring that taking a dual step keeps y^{k+1} feasible. We derive a bound for the dual variable in section 4.5. We remark that this is a dual update at step k, which has a dependence on parameters (augmented Lagrangian parameter λ and smoothing parameters β) of step $k+1$. This relation originates from the convergence proof of CGAL [56].

In practice, we ignore the dual diameter check and simply update dual variable y freely. We observed that this relaxation is numerically stable from results in 5, as well as reports from authors of CGAL [56]. Dual diameter is used more for the theoretical justification than for the algorithmic perspective. We also note that our dual bound derived in section 4.5, might be very large because of weight update. Although scaling would help the bound, the Lagrange multiplier would relatively be scaled by the same amount so that the relative scale would remain the same. Therefore it is of little practical concern for our algorithm.

4.4.3 ADMM vs CGAL

The convergence properties of ADMM are well studied in the literature [63]. We refer to [64] and references therein for a proof of $O(1/\epsilon)$ convergence rate in the general case. In ADMM the most computationally challenging step is the projection to the positive semi-definite cone which requires full eigenvalue decomposition at the cost of $O(n^3)$. Competing algorithms also have bottleneck steps which are equally computationally demanding e.g., [21, 22, 24] eigenvalue decomposition for gradient computation or one sweep over the matrix in [10]. In practice it is possible to replace full eigenvalue decomposition with partial decomposition to save computational effort. However, in our problem, ADMM tends to

create dense eigenvalues initially, which requires about half of the decomposition. This behaviour of ADMM makes partial decomposition either too slow or too inaccurate to be deployed in the algorithm instead of full decomposition without any prior bound on the distribution of eigenvalues. ADMM also has large memory requirements, demanding storage of five $n \times n$ matrices. As a consequence, ADMM may not be the appropriate choice of algorithm for very high dimensional problems like applications in genetics and signal processing. That being said, ADMM still has favourable numerical properties. Primal and dual residual based early termination often saves a lot of computational time. In addition, warm starts using the solution of previous re-weighted optimization often work well with ADMM especially if the problem is solved to moderate accuracy.

CGAL converges provably with $O(1/\epsilon^2)$ rate with respect to both objective function and feasibility measures [56]. In CGAL the main computationally heavy step is the linear minimization oracle. This requires an eigenvalue computation which can be performed in $O(n^2)$, an order of magnitude smaller than ADMM. Moreover, throughout the iterations, CGAL does not produce very dense eigenvalues, and thus it is possible to store the primal variable X in a low-rank fashion to reduce memory requirements. Then, one needs to be careful with the CG-Step (on primal variable X) of the algorithm which amounts to a symmetric rank-1 update. The authors of CGAL proposed this idea for the SDP template in a follow-up paper [65]. They use the *Nyström sketch* method to construct provably accurate low-rank solutions [66]. We leave this computational extension for further research. Additionally, further savings in memory usage are possible by resorting to simpler methods in the computation of eigenvectors. However, since eigenvector computations are carried out iteratively, simpler methods often require a larger number of iterations to get the same precision compared to more involved methods like the Lanczos method [67, 68]. Thus, depending on the problem size, an appropriate eigenvector solver can be chosen. However, unlike ADMM, warm starts often do not save significant computational effort, and early stop criteria either terminate the algorithm early or cause it to reach the maximum number of iterations allowed.

4.5 Analysis of PCA Sparsified

In this section, we analyze the convergence properties of PCA Sparsified. We show that any limit point of the reweighted optimization algorithm is a stationary point of the surrogate function, and the distance between successive iterates tends to zero. A discussion of convergence of distance in X is deferred to Appendix D. Furthermore, the objective function value of the surrogate function decreases monotonically to a limit. Additionally, we derive an upper bound for the dual variable corresponding to the variance constraint.

4.5.1 Convergence in Objective Function Value

We firstly show convergence in objective function value and distance between consecutive iterates.

Theorem 1. *Let $\{(X^{(t)}, U^{(t)})\}$ be the sequence generated by PCA Sparsified algorithm. Then the sequence $\{g(U^{(t)})\}$ converges and distance between successive U's converge to zero, i.e. $||U^{(t)} - U^{(t+1)}||_F \to 0$.*

Proof. We begin with the equivalent formulation given in (4.8).

$$\begin{array}{ll} \min\limits_{X,U} & g(U) = \sum_{i=1}^{n} \sum_{j=1}^{n} \log(\epsilon + u_{ij}) \\ \text{st.} & (X, U) \in C. \end{array} \tag{4.36}$$

The gradient and the Hessian of the surrogate function are given as

$$\begin{aligned} [\nabla g(U)]_{ij} &= \frac{1}{\epsilon + u_{ij}} \\ [\nabla^2 g(U)]_{ij,kl} &= \begin{cases} \dfrac{-1}{(u_{ij} + \epsilon)^2} & if (i,j) = (k,l) \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{4.37}$$

Since we have a trace constraint in our problem, all x_{ij}'s are bounded above by 1. Then we can add $u_{ij} \leq 1$ to the formulation without any loss of information.

As a consequence, the objective function $g(U)$ has the strong concavity property, ($-g(U)$ is strongly convex), and we can write the following inequality

$$\nabla^2 g(U) \preceq \frac{-1}{(1+\epsilon)^2} I_{n^2}. \tag{4.38}$$

Then, the gradient inequality becomes [69]

$$g(V) \leq g(U) + \langle \nabla g(U), V - U \rangle - \frac{1}{2(1+\epsilon)^2} ||V - U||_F^2. \tag{4.39}$$

Applying our iterative reweighted optimization algorithm, the objective values of the surrogate function throughout the iterations are connected through the following inequalities

$$\begin{aligned} g(U^{(t+1)}) &\leq g(U^{(t)}) + \langle \nabla g(U^{(t)}), U^{(t+1)} - U^{(t)} \rangle - \frac{1}{2(1+\epsilon)^2} ||U^{(t+1)} - U^{(t)}||_F^2 \\ &\leq g(U^{(t)}) - \frac{1}{2(1+\epsilon)^2} ||U^{(t+1)} - U^{(t)}||_F^2. \end{aligned} \tag{4.40}$$

The second line follows from the fact that $(X^{(t+1)}, U^{(t+1)}) \in \underset{(X,U)\in C}{\arg\min} \langle \nabla g(U^{(t)}), U \rangle$ and $(X^{(t)}, U^{(t)})$ is a feasible point. Now define the sequence

$$\mathcal{D}_t = \textstyle\sum_{i=0}^{t} ||U^{(i+1)} - U^{(i)}||_F^2, \tag{4.41}$$

which is non-decreasing. Moreover, using the previously stated gradient inequality we can write

$$g(U^{(t+1)}) \leq g(U^{(0)}) - \frac{1}{2(1+\epsilon)^2} \mathcal{D}_t \quad \forall t \geq 1. \tag{4.42}$$

It is known by the construction of $g(U)$ that it is the sum of monotone increasing concave functions. Then, one has

$$\begin{aligned} & n^2 \log(\epsilon) \leq g(U^{(t+1)}) \leq g(U^{(0)}) - \frac{1}{2(1+\epsilon)^2} \mathcal{D}_t \\ \implies \quad & \mathcal{D}_t \leq 2(1+\epsilon)^2 (g(U^{(0)}) - n^2 \log(\epsilon)). \end{aligned} \tag{4.43}$$

Thus, the sequence $\mathcal{D}_t$ is bounded. Therefore it converges. As a consequence we

have

$$\lim_{t\to\infty} ||U^{(t+1)} - U^{(t)}||_F^2 = 0 \tag{4.44}$$

This proves that the distance between successive iterates of $U^{(t)}$ generated by PCA Sparsified converges to 0. We remark that this also proves that sequence $g(U^{(k)})$ also converges since it decreases monotonically and is bounded below. □

We remark that in our proof, we constructed a lower bound for the objective value using a monotonicity argument. A more elegant way to construct a lower bound would be using the fact that $g(U)$ is a continuous function, thanks to the ϵ stability term, on a compact set. Hence it attains its minimum value, say $g(U^*)$.

4.5.2 Stationarity of Limit Points

Now we establish the stationarity property of PCA Sparsified.

Theorem 2. *Let $\{(X^{(t)}, U^{(t)})\}$ be the sequence generated by PCA Sparsified algorithm. Then any limit point of this sequence is a stationarity point of (4.8).*

Proof. Consider $(\bar{X}, \bar{U})$ any limit point of $(X^{(t)}, U^{(t)})$, then there exists a subsequence $(X^{(t_s)}, U^{(t_s)})$ that converges to $(\bar{X}, \bar{U})$. Now assume $(\bar{X}, \bar{U})$ is not stationary, hence $\exists (Y, Z) \in C$ such that

$$-\delta = \langle \nabla g(\bar{U}), Z - \bar{U} \rangle < 0. \tag{4.45}$$

Since $g(U^{(t)})$ converges, $g(U^{(t_s)})$ converges to the same limit. Hence, there exists $M_1 \in \mathbb{N}$ such that

$$g(U^{(t_s)}) - g(\bar{U}) < \tfrac{\delta}{4} \quad \forall\, s \geq M_1. \tag{4.46}$$

As $g(U^{(t)})$ is a monotone non-increasing sequence one has $g(U^{(t)}) \geq g(\bar{U})\, \forall t \in \mathbb{N}$. Also, the surrogate function $g(U)$ is continuously differentiable everywhere, and hence its gradient $\nabla g(U)$ is a continuous function. Then $\langle \nabla g(U^{(t_s)}), Z - U^{(t_s)} \rangle$

converges to $-\delta = \langle \nabla g(\bar{U}), Z - \bar{U} \rangle$. Therefore, there exists $M_2 \in \mathbb{N}$ such that

$$\begin{aligned} & |\langle \nabla g(U^{(t_s)}), Z - U^{(t_s)} \rangle + \delta| < \tfrac{\delta}{2} \quad \forall\, s \geq M_2 \\ \implies \quad & -\tfrac{3\delta}{2} < \langle \nabla g(U^{(t_s)}), Z - U^{(t_s)} \rangle < -\tfrac{\delta}{2} \quad \forall\, s \geq M_2. \end{aligned} \tag{4.47}$$

Now pick $s \geq M = \max\{M_1, M_2\}$. Combining inequalities (4.46) and (4.47) gives;

$$\begin{aligned} & g(U^{(t_s+1)}) \leq g(U^{(t_s)}) + \langle \nabla g(U^{(t_s)}), U^{(t_s+1)} - U^{(t_s)} \rangle \leq g(U^{(t_s)}) - \tfrac{\delta}{2} \\ \implies \quad & g(\bar{U}) \leq g(U^{(t_{s+1})}) < g(U^{(t_s)}) - \tfrac{\delta}{2} < g(\bar{U}) - \tfrac{\delta}{4}, \end{aligned} \tag{4.48}$$

which is a contradiction. In conclusion, every limit point of PCA Sparsified is a first-order stationary point of the surrogate function g over the optimization domain C. □

We note that in [17], a similar result was established for maximizing a convex function over a compact set in a more general setting without assuming a strong concavity like property with a direct proof. Here we give a proof by contradiction for the same problem while strong concavity allows us to also show that the distance between consecutive iterates goes to zero.

4.5.3 Frank-Wolfe Gap Convergence

Although in the previous section, we show that any limit point of the sequence generated by PCA Sparsified is stationary, our proof does not give any sense of speed of convergence to a stationary point. In this section, we analyze the convergence speed in terms of the stationarity property of PCA Sparsified.

Recall that our algorithm generates a sequence of $(X^{(t+1)}, U^{(t+1)})$ by iteration given by the CG iteration (4.10). Now let us define the following quantity

$$h_t = \max_{(Y,Z) \in C} \langle -\nabla g(U^{(t)}), Z - U^{(t)} \rangle \tag{4.49}$$

this quantity measures how far away the current point is from being stationary

because any local minimum has to satisfy (4.9). This is often referred to as Frank-Wolfe[2] gap at iteration t, which is commonly used in the analysis of CG-type algorithms [70, 71]. In [72], it was shown that the minimum FW gap encountered by the FW algorithm applied a concave function over a compact convex set up to iteration t is bounded by $K/(t+1)$, where K is some constant related to initial guess. This theorem immediately applies to our algorithm and optimization framework. For completeness, we include an adapted proof here.

Theorem 3. *Let $\{(X^{(t)}, U^{(t)})\}$ be the sequence generated by PCA Sparsified algorithm. Then, the minimum FW gap defined as $\hat{h}_t = \min_{0\le k\le t} h_k$ satisfies*

$$\hat{h}_t \le \frac{g(U^{(0)}) - g(U^*)}{t+1} \tag{4.50}$$

where $g(U^*) = \min_{(X,U)\in C} g(U)$ is the global minimum value of the function g over compact convex set C.

Proof. We start by rewriting the equation (4.40) and applying the definition (4.49).

$$\begin{aligned} g(U^{(t+1)}) &\le g(U^{(t)}) + \langle \nabla g(U^{(t)}), U^{(t+1)} - U^{(t)} \rangle - \frac{1}{2(1+\epsilon)^2} ||U^{(t+1)} - U^{(t)}||_F^2 \\ g(U^{(t+1)}) &\le g(U^{(t)}) - h_t - \frac{1}{2(1+\epsilon)^2} ||U^{(t+1)} - U^{(t)}||_F^2 \end{aligned} \tag{4.51}$$

The second line follows from the construction of $(X^{(t+1)}, U^{(t+1)})$. Summing up the inequalities inductively gives

$$\begin{aligned} g(U^{(t+1)}) &\le g(U^{(0)}) - \textstyle\sum_{i=0}^{t} h_t - \frac{1}{2(1+\epsilon)^2}\mathcal{D}_t \\ &\le g(U^{(0)}) - (t+1)\hat{h}_t - \frac{1}{2(1+\epsilon)^2}\mathcal{D}_t \end{aligned} \tag{4.52}$$

[2]The conditional gradient (CG) algorithm is also known as Frank-Wolfe (FW) algorithm

Rearranging terms in the second inequality (4.52) gives;

$$\begin{aligned}(t+1)\hat{h}_t &\leq g(U^{(0)}) - g(U^{(t+1)}) - \frac{1}{2(1+\epsilon)^2}\mathcal{D}_t \\ \hat{h}_t &\leq \frac{g(U^{(0)}) - g(U^*)}{k+1}\end{aligned} \tag{4.53}$$

The last line of (4.53) gives precisely the inequality (4.50) where we used both the non-negativity of $\mathcal{D}_t$ and the global minimality of the $g(U^*)$. □

Remark that global optimum in (4.50) can be changed with any lower bound on the sequence of objective values that are generated by PCA Sparsified.

4.5.4 Dual Bounds

We start by constructing the dual problem corresponding to PCA Sparsified. Replacing the ℓ_1 norm with its dual characterization and moving the variance constraint into the objective function by dualizing we get

$$\begin{aligned}\max_{y\geq 0}\min_{X} \quad & \max_{||V||_\infty \leq 1} \langle V, W \circ X\rangle - \langle y, \mathrm{tr}(AX) - b\rangle \\ \text{st.} \quad & \mathrm{tr}(X) = 1 \\ & X \succeq 0.\end{aligned} \tag{4.54}$$

By Sion's minimax theorem [73], we can interchange min and max to get the following dual problem

$$\begin{aligned}\max_{||V||_\infty \leq 1, y\geq 0} yb + \quad & \min_X \langle W \circ V - yA, X\rangle \\ & \text{st. } \mathrm{tr}(X) = 1 \\ & X \succeq 0.\end{aligned} \tag{4.55}$$

The solution of the inner minimization problem is given by the minimum eigenvector, i.e., $X = v_{\min}(W \circ V - yA)v_{\min}(W \circ V - yA)^T$. We arrive at the following

form for the dual problem

$$\begin{array}{ll} \max & yb + \lambda_{\min}(W \circ V - yA) \\ \text{st.} & ||V||_\infty \leq 1 \\ & y \geq 0. \end{array} \tag{4.56}$$

Consider the first step of the algorithm PCA Sparsified where all weights are equal and scaled to be equal to one. Then we can construct lower bounds for the dual problem by finding feasible points. For instance, taking $y = 1/d_{\max}, V = A/d_{\max}$ where $d_{\max}$ as defined before in section 4.3 gives a lower bound on the objective value $d^* \geq b/d_{\max}$. Let y^* and V^* denote optimal dual variables then we have

$$\tfrac{b}{d_{\max}} \leq by^* + \lambda_{\min}(V^* - y^*A). \tag{4.57}$$

Note that $-\lambda_{\min}(X)$ is a subdifferentiable convex function, and $-v_{\min}(X)v_{\min}(X)^T$ is a subgradient [74]. Around $U = 0$ and $y > 0$, $-\lambda_{\min}(V^* - y^*A)$ has a subgradient $-v_{\max}(A)v_{\max}(A)^T$. So, the inequality above becomes

$$\tfrac{b}{d_{\max}} \leq by^* - y^*\lambda_{\max}(A) + \langle vv^T, V\rangle. \tag{4.58}$$

Where $v = v_{\max}(A)$. By using Hölder's inequality we can bound the inner product term on the right hand side since the ℓ_∞ norm of V is bounded above by one. Then, we get the following upper bound on the dual variable

$$\begin{array}{rl} & \dfrac{||vv^T||_1 - b/d_{\max}}{\lambda_{\max}(A) - b} \geq y^*, \\ \implies & \dfrac{\lambda_{\max}(A) - b}{||vv^T||_1 - b/d_{\max}} \leq \dfrac{1}{y^*}. \end{array} \tag{4.59}$$

Remark that $b/d_{\max}$ can be sharpened with eigenshifts and variance adjustments for better bounds, if there is no off-diagonal element that is equal to $d_{\max}$. This dual bound may be weak for some instances of PCA Sparsified because of the feasible point we constructed. Better dual lower bounds can be computed cheaply using $U = \text{sign}(A)$ and performing a golden section search over y. Additionally, this bound can be adapted to the weighted ℓ_1 norm in a similar fashion for CGAL dual bound.

We showed in subsection 4.3.2 that DSPCA and PCA Sparsified are equivalent up to a Lagrange multiplier. Furthermore, the multiplicative inverse of the Lagrange multiplier corresponds to the penalty term for the DSPCA formulation. Therefore the bound derived above serves as a lower bound for the penalty term for the equivalent DSPCA formulation. Now, we invoke the safe feature elimination theorem presented in [8] for the ℓ_0 penalized model. This pre-processing method was applied to the SDP model in [10] as a convex approximation to the original problem. Since our problem at the first step and the DSPCA formulations are equivalent we also apply the same preprocessing method. However, since we do not have the precise penalty term (multiplicative inverse of Lagrange multiplier) prior to solving the problem, instead we use the lower bound derived in (4.59). Finally, in Appendix E we present a similar pre-processing idea derived from the first step of the algorithm i.e., pre-processing for DSPCA.

4.6 Multiple Components

PCA Sparsified was designed to extract a single sparse component. Multiple components can be extracted by deploying a deflation scheme to get a sparse decomposition of the covariance matrix as described in 3.5. By iteratively applying PCA Sparsified on the new matrix and projection on orthogonal subspace, second and subsequent loadings and components can be computed.

A more natural and efficient approach to this problem could be obtained by solving for all k loadings at the same time. Since we are using an SDP approach, our model can be modified to compute all k loadings at the same time by replacing the "spectraplex" constraint set $\mathcal{X} = \{X \mid \mathrm{tr}(X) = 1, X \succeq 0\}$ with the so-called "Fantope" $\mathcal{F}_k = \{X \mid \mathrm{tr}(X) = k, I \succeq X \succeq 0\}$. This idea is proposed in [24] for the ℓ_1 norm penalized model. The Fantope model for multiple components can also be used in our approach. However, in practice, if the underlying matrix does not have a sparse decomposition, this leads to sparsity in the matrix variable X, yet the loadings extracted from X are often dense. Another issue with this approach is rank relaxation. As we discuss next, PCA Sparsified often

generates rank-one solutions. However, for the Fantope model, achieving an exact rank-k optimal solution is a bigger problem. Taking the top k eigenvectors and truncating by sparsity level could lead to substantial losses in variance explained when the solution Fantope approach computes does not have an approximately rank-k decomposition with sparse loadings. Hence, depending on the application, a suitable approach can be deployed.

As stated before in 3.5, unlike PCA, in SPCA orthogonality of the loadings and the uncorrelation of the components are often lost. If Fantope is used instead of the spectraplex, orthogonality of the loadings is assured since eigenvectors of a symmetric matrix are orthogonal (assuming they are sparse and the matrix is rank k for sparse decomposition) but the uncorrelation of the components remains an issue. Hence, to measure the performance of the algorithms in multiple components setting, adjusted variance (3.20) or (3.21) should be used.

In [28], the uncorrelation of the components or orthogonality of the loadings is enforced for the multiple components exact SPCA problem. This feature can also be extended to PCA Sparsified by adding linear constraints to explicitly enforce the orthogonality of loadings or the uncorrelation of the components. We leave this extension as further research. In practice PCA Sparsified, having only two linear constraints, often generates rank-one matrices. Adding extra linear constraints may lead to the loss of this property. Indeed, by optimality conditions, we may expect X to be rank one. Assume that X is a minimizer of (4.12), then there exists a subgradient G of $||W \circ X||_1$ such that [75]

$$\langle G, Y - X \rangle \geq 0 \quad \forall\, Y : \mathrm{tr}(AY) = b, \mathrm{tr}(Y) = 1, Y \succeq 0, \tag{4.60}$$

which implies that X is also a minimizer of the following problem

$$\begin{array}{ll} \min\limits_{Z} & \langle G, Z \rangle \\ \text{st.} & \mathrm{tr}(AZ) = b \\ & \mathrm{tr}(Z) = 1 \\ & Z \succeq 0. \end{array} \tag{4.61}$$

Since this is an SDP with linear objective function and two linear constraints, there exists a rank one optimal solution [76]. Thus, if the solution of problem (4.61) is unique, X is necessarily rank-one.

4.7 Sufficient Conditions for Theoretical Guarantees

In this section, we analyze further theoretical properties of PCA Sparsified.

4.7.1 Local Minimum and Rank-one Connection

PCA Sparsified algorithm iteratively minimizes log-sum penalty function locally by the first-order stationarity principle given in (4.9). Consider now the second-order stationarity condition [41]

$$(Z-U)^T\nabla^2 g(U)(Z-U)\geq 0 \quad \forall\,(Y,Z)\in C \text{ s.t. } \nabla g(U)^T(U-Z)=0. \tag{4.62}$$

The condition given in (4.62) implies that, if (X,U) is a local minimum of g, then it must be strictly stationary because g strongly concave. To the contrary assume that there exists a (Y,Z) in the intersection of the orthogonal subspace of $\nabla g(U)$ and set C. Then by Taylor expansion $\exists\eta\in[Z,U]$ s.t.

$$\begin{aligned} g(Z) &= g(U)+\nabla g(U)^T(Z-U)+\tfrac{1}{2}(Z-U)^T\nabla^2 g(\eta)(Z-U) \\ &\leq g(U)-\frac{||Z-U||_F^2}{2(1+\epsilon)^2} \end{aligned} \tag{4.63}$$

where we used strong concavity property (4.38) and orthogonality to the gradient. Last inequality in (4.63) shows that (Y,Z) achieves a smaller objective value. We can take (Y,Z) arbitrarily close to (X,U) to show that (X,U) is indeed not a local minimum.

If it is possible to compute the entire optimal solution set to (4.12), then

even when the point (X, U) is stationary but not strictly stationary, we can find descent directions to further improve the objective. However, this is a difficult task. Instead, we could look for dual certificates that will guarantee that a point (X, U) is second-order stationary or first-order strict stationary. We propose one such condition;

Proposition 1. *If there exists $y \in \mathbb{R}_+$ and $F \in \mathbb{R}^{n\times n}$ obeying following conditions;*

$$\begin{aligned} &||F||_\infty \leq 1 \\ &F_{ij}X_{ij} = 0 \\ &H = W \circ \operatorname{sign}(X) + W \circ F - yA \\ &\lambda_n(H) < \lambda_{n-1}(H) \\ &X = v_n(H)v_n(H)^T \end{aligned} \tag{4.64}$$

Then X is a second-order stationary and first-order strict stationary point of the surrogate function g, i.e. unique minimizer of the linearization of g around itself.

Conditions in (4.64) can be interpreted as follows; F is a subgradient of $||X||_1$, matrix H is the placeholder for a subgradient of the Lagrangian function corresponding to (4.12) after dualizing the affine constraint $\operatorname{tr}(AX) = b$ as in (4.13). The eigenvalue and vector conditions imply that there is a unique minimizer of the Lagrangian over the spectraplex set.

Proof. Consider any feasible point Y that is not equal to X. We start by writing the Lagrangian function

$$\begin{aligned} ||W \circ Y||_1 &\geq ||W \circ Y||_1 - y(\operatorname{tr}(AY) - b) \\ &\geq ||W \circ X||_1 - y(\operatorname{tr}(AX) - b) + \langle W \circ sign(X) + W \circ F - yA, Y - X\rangle \\ &> ||W \circ X||_1 \end{aligned} \tag{4.65}$$

In the second line, we used subgradient $W \circ (sign(X) + F) \in \partial||W \circ X||_1$ and its underestimating property. In the third line, we use the hypothesis that X is feasible, $\operatorname{tr}(AX) = b$, and X is the unique minimizer of that particular choice of subgradient over the spectraplex set.

This also implies that (X, U) pair defined by $u_{ij}^{(t+1)} = |x_{ij}^{(t+1)}|$ is strictly stationary and hence unique minimizer of the equivalent formulation (4.10). If (Y,Z) pair is another minimizer of (4.10), $||W \circ Y||_1 = ||W \circ X||_1$ is a contradiction, given that $Y \neq X$. If $Y = X$, then this leads to another contradiction, $u_{ij}^{(t+1)} = |x_{ij}^{(t+1)}|$ is the unique minimizer of the problem whenever X is fixed. □

Conditions given in (4.64) are sufficient for the unique local minimality of (X, U) for the linearized problem which was conjectured in the last part of section 4.6 for rank-one recovery.

4.7.2 Strict Local minimum of the surrogate function

In previous sections, we showed that if a computed point (X, U) is second-order stationary then X is rank-one. However, it may still not be a local minimum, which is desirable because if we can improve the objective, it could still enhance the variance explained and sparsity. For this purpose, we adopt a first-order sufficient condition [77] which can be checked by solving an auxiliary SDP model. Remark that the local minimality property may not be very meaningful as this function may have a lot of bad local minima because the PCA Sparsified algorithm minimizes a concave function over a convex compact set.

Proposition 2. *X is a strict local min if there exist no non-trivial solutions to the following system, i.e. if the only feasible solution to the following system is $(P, Q) = 0$*

$$\begin{aligned} P_{ij} &\leq Q_{ij} \quad \forall\, i,j \in S_+ \\ -P_{ij} &\leq Q_{ij} \quad \forall\, i,j \in S_- \\ \operatorname{tr}(P) &= 0 \\ \operatorname{tr}(AP) &= 0 \\ R^T P R &\succeq 0 \\ \langle W, Q \rangle &\leq 0 \end{aligned} \tag{4.66}$$

where R matrix consists of eigenvectors of X with eigenvalue 0, S_+ denotes set of indices where $x_{ij} = u_{ij}$ and S_- denote of indices where $-x_{ij} = u_{ij}$.

Conditions in (4.66) can be interpreted as infeasibility of a descent direction in a non-strict fashion.

Proof. Infeasibility of the system (4.66) except for zero (P, Q), implies that (X, U) is a strict local minimum of the log-sum penalty. To the contrary assume that there exists sequence of points $(Y_n, V_n) \in C$ such that

$$\begin{aligned} (Y_n, V_n) &\to (X, U) \\ g(V_n) &< g(U) \quad \forall\, n \end{aligned} \tag{4.67}$$

Consider the following sequence of matrices on unit ℓ_1 ball

$$\begin{aligned} Q_n &= \frac{V_n - U}{||V_n - U||_1} \\ P_n &= \frac{Y_n - X}{||Y_n - X||_1} \end{aligned} \tag{4.68}$$

Since these sequences are on a compact set, we can consider a subsequence with limits $P = \lim_{k\to\infty} P_{n_k}, Q = \lim_{k\to\infty} Q_{n_k}$. To simplify the notation we will assume original sequences are convergent. Then we have

$$\begin{aligned} \frac{g(V_n) - g(U)}{||V_n - U||_1} &< \tfrac{1}{n} \\ \frac{g(V_n) - g(U)}{||V_n - U||_1} &< 0 \end{aligned} \tag{4.69}$$

Taking limits gives

$$\begin{aligned} \langle \nabla g(U), Q \rangle &\leq 0 \\ \langle W, Q \rangle &\leq 0 \end{aligned} \tag{4.70}$$

In the second line, we use the fact that weights are determined by the gradient in our algorithm. Similarly, we can apply the same reasoning to get

$$\begin{aligned} P_{ij} &\leq Q_{ij} \quad \forall\, i, j \in S_+ \\ -P_{ij} &\leq Q_{ij} \quad \forall\, i, j \in S_- \\ \operatorname{tr}(P) &= 0 \\ \operatorname{tr}(AP) &= 0 \\ R^T P R &\succeq 0 \end{aligned} \tag{4.71}$$

For the first inequality, assume to the contrary that $\exists\, i, j \in S_+$ such that $P_{ij} > Q_{ij}$. Then for sufficiently large n, strict inequality would be preserved so we get

$$\begin{aligned} \left[\frac{Y_n - X}{||Y_n - X||_1}\right]_{ij} &> \left[\frac{V_n - U}{||V_n - U||_1}\right]_{ij} \\ \left[\frac{Y_n - X}{||V_n - U||_1}\right]_{ij} &> \left[\frac{V_n - U}{||V_n - U||_1}\right]_{ij} \\ [Y_n]_{ij} &> [V_n]_{ij} \end{aligned} \tag{4.72}$$

which leads to a contradiction since (Y_n, V_n) is not feasible to (4.8). Second line follows from the relation $||Y_n - X||_1 \geq ||V_n - U||_1$. While this is not necessarily true, we can assume that $[V_n]_{ij} = |\,[Y_n]_{ij}\,|$, without loss of generality, because if it is not the case, we can take this new sequence, which will follow assumptions of (4.67). Third line follows from $x_{ij} = u_{ij}\ \forall\, i, j \in S_+$. The same proof can be applied to show that the second inequality in (4.71) also follows. Furthermore, a similar line of reasoning can be applied to show that (P, Q) will satisfy the third and the fourth constraints because (Y_n, V_n) is always feasible and equality is preserved under the limit operation. The LMI constraint follows similarly using congruence transform and the fact that R matrix is an orthogonal complement of X

$$\begin{aligned} Y_n \succeq 0 \implies & R^T Y_n R \succeq 0 \\ \implies & R^T (Y_n - X) R \succeq 0 \\ \implies & R^T \frac{Y_n - X}{||Y_n - X||_1} R \succeq 0 \\ \implies & R^T P R \succeq 0. \end{aligned} \tag{4.73}$$

□

4.8 Implementation and Computational Tactics

In this section, we briefly discuss the implementation details of CGAL and ADMM algorithms.

For both CGAL and ADMM, scaling the weight matrix W leads to performance gains allowing the use of simple stability parameters that work well in most cases.

At iteration $(t+1)$, given the optimal solution of the previous iteration $X^{(t)}$, we first scale $X^{(t)}$ by its maximum value. Then we use the correction term $\epsilon = \frac{0.5}{s}$ to find the weights for the next iteration. Finally, we scaled W by its largest value to normalize it. Our numerical results and operator norm computed for W motivate this scaling scheme. The correction term is motivated by the structure of a s-sparse normalized vector. The smallest possible value for the largest term of a s-sparse vector x is $\frac{1}{\sqrt{s}}$. For the matrix variable $X = xx^T$ this becomes $\frac{1}{s}$ in the diagonal. This choice of parameter promotes features that have a large value in the SDP solution while still allowing discovery of other features that may be given small values.

Poorly chosen stability parameters often lead to similar behaviour in both ADMM and CGAL. If a very small stability parameter value is chosen, the matrix variable X tends to stay high rank throughout iterations and produce poor quality solutions for the original non-convex problem. Conversely, if a very large stability parameter value is chosen, matrix variable X generally tends to have low-rank solution throughout iterations but requires a large number of reweighted optimization iterations to produce high quality solutions.

For CGAL, we scaled the Frobenius norm of A to be 1. In fact, without scaling, CGAL performed very poorly and even failed the eigenvector computation using ARPACK [78]. Later we changed the eigenvector computation with LAPACK dsyevr subroutine which turned out to be faster and more robust [79]. After scaling, augmented Lagrangian parameter choice $\rho_0 = 1$ worked well and did not require any parameter search. Also, we omitted the dual update since the bound on dual diameter becomes large after the first reweighted iteration. We did not encounter any example where the dual variable diverges. For the initial smoothing parameter, we choose $\beta_0 = \frac{0.5}{n}$. The choice of smoothing parameter often affected the behaviour of the algorithm. A small value for the smoothing parameter often resulted in an uncontrolled subgradient sequence, and large values for the smoothing parameter failed to give sparse solutions. Similar to stability parameter choice, we had the following intuition about our choice of smoothing parameter: in the first few steps of the algorithm, X tends to Frobenius norm scaled A. Then, the smoothing parameter for a dense vector of dimension n

should be set proportional to $\frac{1}{n}$.

For ADMM, scaling the problem data did not have a discernible effect on the results. This is most likely due to the implementation of the adaptive augmented Lagrangian parameter which handles problem scale and dual feasibility. Furthermore, ADMM seems to be reasonably robust to initial augmented Lagrangian parameter choice for a similar reason. We chose $\rho = 40$ which was the best in our experience. However, the order of minimization between X_i's changed the stability of the problem. For certain permutations, X_i's diverged to infinity or converged to all zeros. In our implementation, the minimization order is X_2, X_1, and X_3.

Chapter 5

Numerical Experiments

By virtue of the approaches developed for solving the SPCA problem, we present our numerical results in two categories. First, computationally very cheap algorithms that scale easily to large problems are compared, as well as their performance on synthetic problems. Secondly, we focus on PCA Sparsified, which is computationally very demanding. Therefore, we focus on results on small dimensional problems. Our performance criteria in these experiments are variance explained, CPU time, and Wall time of the algorithms.

Computational experiments were conducted on a machine with Intel Core i7-9700K CPU and 128 GB RAM.

5.1 Scalable Algorithms

This section presents extensive numerical experiments on scalable algorithms which are often local models 2.1, i.e. either gradient or greedy type algorithms. Since we consider only local models here, in order to show the effectiveness of the algorithms we consider three types of datasets; synthetic, random, and real. While we keep the dimension of the synthetic datasets small, we work with very

high dimensional datasets in random and real datasets. Our local greedy algorithms developed in 3 are referred to as GD (Gerschgorin Disc) 2, as CCW (Correlation Coordinate Wise) 1 and as FCW (Frobenius Coordinate Wise) 3, respectively. Python implementation of our algorithms is also publicly available [1].

We compared several algorithms numerically including approximate greedy (Path) [8], expectation-maximization (EM) [15], generalized power method (GPower) [16], partial and greedy coordinate-wise (PCW and GCW) [20] and Approximate Newton with line search (GPBB-ls) [14]. Our own algorithms are referred to as GD (Gerschgorin Disc) 2, as CCW (Correlation Coordinate Wise) 1 and as FCW (Frobenius Coordinate Wise) 3, respectively. We implemented the EM algorithm in Python and initialized it with the first principle component which was the default choice. We used available MATLAB codes for PCW, GCW and GPBB-ls, calling them through MATLAB's Python engine. For GPower we used the available Python code from a more recent paper [18], and called it with default parameters, except the number of trials which was set to 100. For the approximate greedy search algorithm, we used the Python code available and initialized it with the variable that has the largest column norm in the covariance matrix. Since we assume that variables are on the same scale and therefore diagonal entries are equal, using the default choice of picking the variable with the largest variance does not make sense in our case. Python implementation of our algorithms is also publicly available.

For each algorithm, we perform a post-processing step and impose a stationarity condition to strengthen the sparsity patterns constructed. Our algorithms apply stationarity conditions to all possible patterns and then choose the best solution, i.e., the stationarity condition is imposed before the last step. EM and GPower algorithms inherently satisfy the stationarity condition. Similarly, it was shown in [20] that PCW and GCW also satisfy this property.

[1] Our local greedy SPCA algorithms are available on: https://github.com/Fatih-S-AKTAS/SPCA.

5.1.1 Synthetic Data Tests

Synthetic data tests were made based on the reference [80] as it is one of the most recent studies in the literature. The data has been generated as follows:

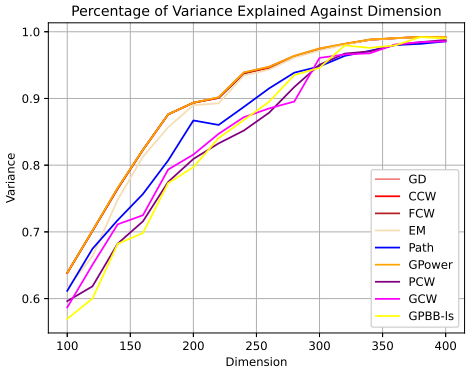
$$\begin{gathered} E_{i,j} \sim \mathcal{N}(0, 0.1) \\ U \sim \mathcal{U}, \quad V \sim \mathcal{V} \\ \Sigma = \text{diag}(\sigma), \quad \sigma_i = 10 - \sqrt{i} \\ S = U\Sigma V^T \\ X = SY^T + E, \end{gathered} \tag{5.1}$$

where $\mathcal{U}$ and $\mathcal{V}$ are orthonormal matrices with the additional sparsity property enforced on $\mathcal{U}$, S and Y are low-rank matrices.

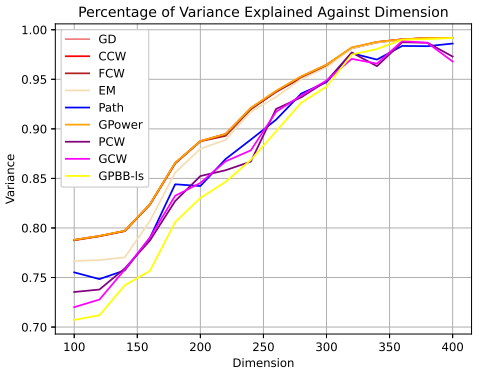
In Figures 5.1d & 5.1b, the dimensions of the generated data matrix X vary from $\mathbb{R}^{100\times100}$ to $\mathbb{R}^{400\times400}$ with increments of 20 in each step. For each increased dimension, the component size is also increased by one to check the explained variance when the dimension gets larger, starting at $rank(S) = 16$. To smooth the randomness of the algorithms we have generated 30 tests and taken the average of the trials for each algorithm.

In addition, we have tested the algorithms under a slightly different setup by generating under-determined systems, i.e., a data matrix with a small number of observations. Dimensions of the matrix X vary from $\mathbb{R}^{10\times100}$ to $\mathbb{R}^{40\times400}$ with increments of 2 and 20 for the number of observations and variables, respectively under this setup. Also, the starting rank of S is reduced to 8 as the construction of S does not allow the rank to be larger than 10. In different experiments, we have tested our algorithms on under-determined systems as they are the more common data types encountered in social networks and genome datasets. The results obtained using under-determined synthetic data are reported in Figures 5.1c & 5.1a.

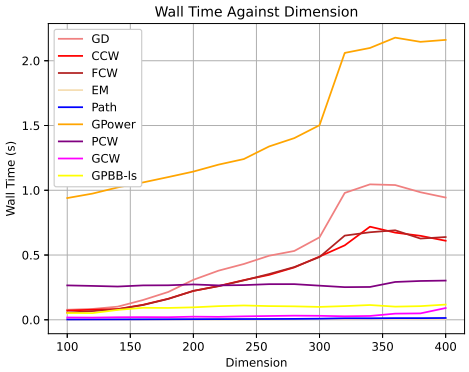
In Figures 5.1d & 5.1b the trade-off between the run-time of our algorithm and the variance explained is shown. All three of our algorithms are competitive with



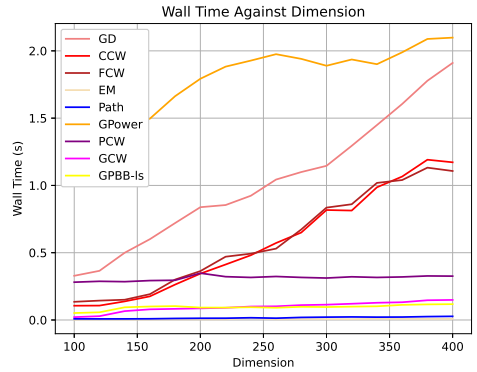
(a) Proportion of Variance Explained with First PC on under-determined Synthetic Data



(b) Proportion of Variance Explained with First PC on Square Synthetic Data



(c) Run Time for the first PC on under-determined Synthetic Data



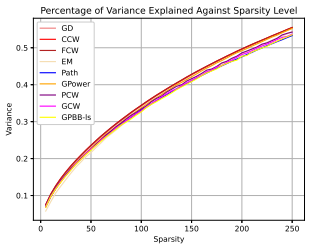
(d) Run Time for the first PC on Square Synthetic Data

Figure 5.1: Run Time and Variance Explained on Synthetic Datasets

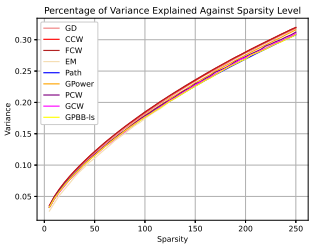
the other algorithms. Furthermore, In Figures 5.1c and 5.1a our algorithms start to have a slight edge in terms of variance explained when the under-determined system gets larger. In terms of variance explained, GD algorithms outperform both CCW and FCW uniformly. CCW and FCW perform similarly while CCW does slightly better. Furthermore, even the FCW algorithm uniformly outperforms other algorithms except for the GPower algorithm. Finally, the GD algorithm does slightly better than GPower.

5.1.2 Random Datasets

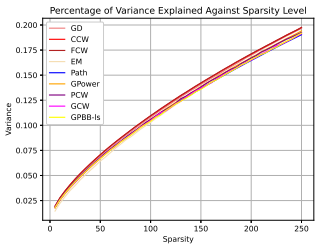
We randomly generated a data matrix $B \in \mathbb{R}^{m \times n}$ from a Gaussian distribution with zero mean and unit variance ($B_{i,j} \sim \mathcal{N}(0,1)$). Sample size was fixed at $m = 100$, and we considered varying the number of variables for $n = 2000, 5000, 10000$. For computational efficiency, we solved problems with sparsity level $s = 5, 10, .., 250$, and the search was conducted over the 200 variables that have the largest column norm. We repeated each experiment 30 times.



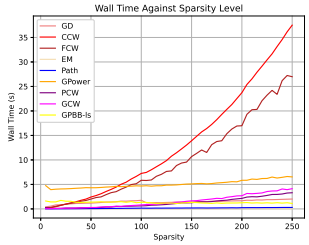
(a) Proportion of Variance Explained with First PC for Changing Sparsity levels, $dim(B) = 100 \times 2000$



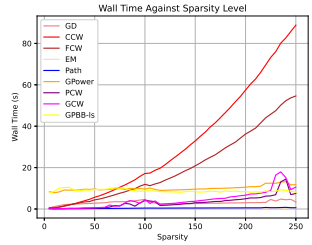
(b) Proportion of Variance Explained with First PC for Changing Sparsity Levels, $dim(B) = 100 \times 5000$



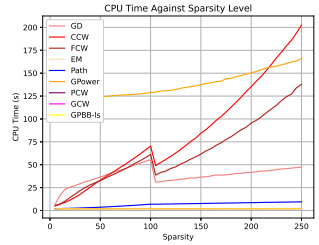
(c) Proportion of Variance Explained with First PC for Changing Sparsity Levels, $dim(B) = 100 \times 10000$



(d) Wall Time of the Algorithms, $dim(B) = 100 \times 2000$



(e) Wall Time of the Algorithms, $dim(B) = 100 \times 5000$



(f) Wall Time of the Algorithms, $dim(B) = 100 \times 10000$

Figure 5.2: Run Time and Variance Explained on Random Datasets

As shown in Figures 5.2d, 5.2a, 5.2e, 5.2b, 5.2f, 5.2c, all three of our algorithms uniformly outperform (in a strict sense) other algorithms in all experiments in terms of proportion of explained variance. Among our proposed algorithms, the GD algorithm is also uniformly dominated by CCW and FCW algorithms in terms of variance explained. However, this result comes at an increased computational burden as the cardinality of the solution desired increases the run time of CCW

and FCW while that of GD remains almost constant at a low level. Hence, GD algorithms seem a computationally attractive way to construct sparse loadings while with more computational resources CCW and FCW might be a better choice for attaining a larger proportion of variance explained. CCW and FCW perform very similarly with FCW having a small advantage in terms of how many times it finds the best solution. In general, our algorithms perform well in high dimensions. As pointed before in chapter 1, in many applications of PCA, dataset shapes are similar, in the sense that they are high dimensional datasets with a small number of data points like in genetics, face recognition, and signal processing.

As a side note, we observed that with a larger number of data points performance of the Path algorithm improves.

5.1.3 Real Datasets

A classical benchmark for the performance of Sparse PCA algorithms is the Pitprops dataset which was initially investigated in [81]. The dataset has 180 data points and 13 variables. We have compared our algorithms on this dataset in terms of variance explained. Figure 5.3 shows the results of the algorithms.

Computational costs of algorithms for this dataset are not reported because they are negligible as the dataset has a small size. Figure 5.3 show that most of the algorithms perform similarly and close to the optimum with small differences. Exact recovery rates of optimum solution of the algorithms for sparsity levels $s = 2, ..., 12$ are summarized in Table 5.1.

Algorithm	GD	CCW	FCW	EM	Path	Gpower	PCW	GCW	GPBB-ls
Recovery	11	10	10	9	11	11	10	10	11

Table 5.1: Exact Recovery Rates of Different Algorithms for Varying Sparsity Levels from 2 to 12 on Pitprops Dataset

AT & T faces dataset is a face recognition dataset that has been used as a benchmark in SPCA studies e.g., in [80]. This dataset has 400 data points and

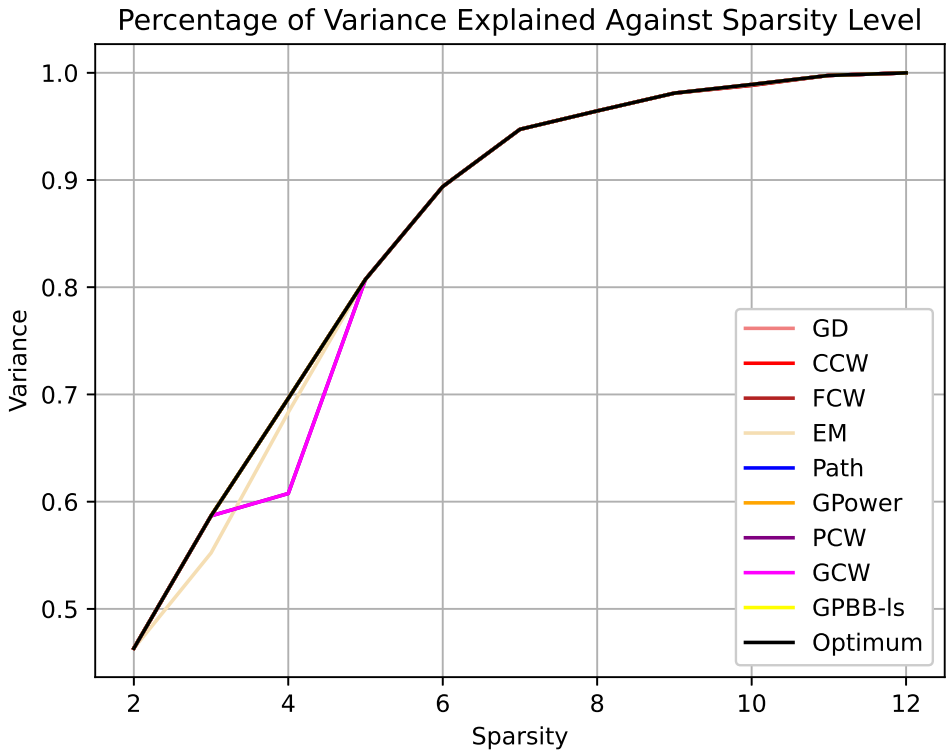
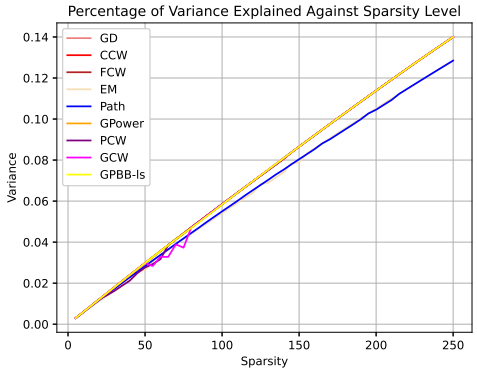


Figure 5.3: Proportion of Variance Explained with First PC for Varying Sparsity Levels on Pitprops Dataset

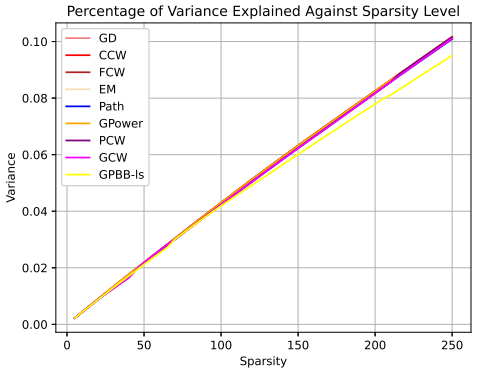
10304 variables. We have compared our algorithms on this dataset in terms of variance explained and CPU-time. Figures 5.4a and 5.4c present the comparison between the competing methods.

Figure 5.4a shows that while most of the algorithms are competitive in terms of variance explained, GPower gives the best performance with a slight edge over other algorithms. GPower does better than all three of our proposed algorithms on this dataset in a non-strict fashion. Other algorithms are either competitive or worse in terms of variance explained. Among our algorithms, CCW performs the best, followed by FCW and GD with a small margin of performance between each of them.

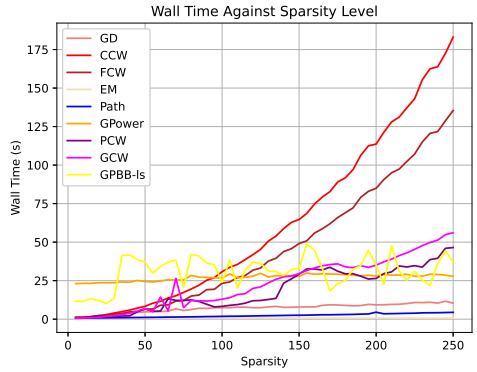
Our final results for this section are on the Arcene dataset which was used in [30] for benchmarking. This dataset has 100 data points and 10000 variables.



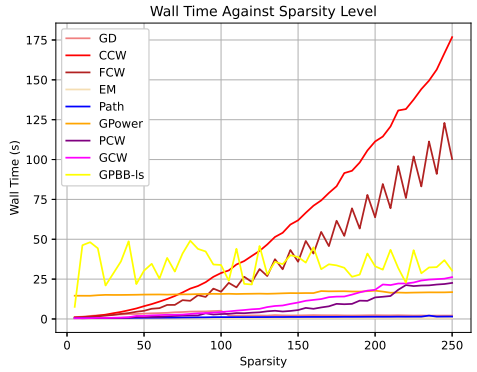
(a) Proportion of Variance Explained with First PC for Changing Sparsity Levels on AT & T Faces Dataset



(b) Proportion of Variance Explained with First PC for Changing Sparsity Levels on Arcene Dataset



(c) Wall Time of Algorithms for Changing Sparsity Levels on AT & T Faces Dataset



(d) Wall Time of Algorithms for Changing Sparsity Levels on Arcene Dataset

Figure 5.4: Run Time and Variance Explained on AT & T and Arcene Datasets

Figures 5.4b and 5.4d summarize the results we obtained.

On the Arcene dataset, FCW and CCW agree on all sparsity levels, while GD has a slight advantage over them. All three of our algorithms overtake other methods in almost all sparsity levels other than GPower, which still remains competitive. On a closer inspection, we see that GPower does better for lower cardinality levels while our algorithms are preferable for higher cardinality levels on this dataset.

It is observed that the variance explained in our methods is superior to the

competing methods in the literature on random and synthetic datasets while being highly competitive in real datasets. Similarly, the CPU time spent by our algorithms, especially Algorithm 2 scales very well with increasing dimensions. Hence, our algorithms are computationally as viable as the Path and GPower algorithms which are considered state-of-the-art. In fact, from our numerical experiments, we see that our GD algorithm is a cheaper alternative to GPower, achieving the same or better level of variance in most of the instances. Furthermore, decreasing the number of random initializations for GPower to match the wall time spent between two algorithms hurt the performance of GPower in terms of variance. We can also see Gershgorin Disc approximation can be seen as a better alternative to randomized initialization adopted in GPower algorithms.

5.1.4 Multiple Components

Our final numerical results are on the performance of the algorithms in computing more than one component. The Deflation Scheme described in Section 3.5 is used to iteratively compute 5 components on AT & T dataset and random dataset of size 100×2000. Moreover, the sparsity level is identical for all components. Figures 5.5a and 5.5b show percentage of explained adjusted variance of the produced components against sparsity level. Figures 5.5c and 5.5d report the computational cost of the algorithms for multiple components.

The performance on randomly generated instances is similar to the one observed in previous experiments. While all algorithms behave similarly, our algorithms have an edge over other algorithms. However, unexpected jittery behavior is observed from the results on the AT & T dataset. As the number of non-zeroes increases, the variance explained does not necessarily increase. Moreover, the performances of the algorithms are not very close until high sparsity levels. We believe that this is related to the nature of the dataset and our scaling procedure. As we centered and scaled the data column-wise, subsequent sparse loadings construct components that are highly correlated with previous components.

5.2 Small Dimensional Problems

This section presents numerical experiments on real and randomly generated small dimensional matrices that illustrate the effectiveness of PCA Sparsified over other existing algorithms. For this purpose, we implemented CGAL and ADMM algorithms in Python using the scientific computing library Scipy [2]. The effectiveness of DSPCA on synthetic datasets was shown in [21] and thus, we omit synthetic dataset results.

We consider local methods against PCA Sparsified. We omit exact approaches here, as they do not scale well even in small-to-medium size data when sparsity gets larger. The algorithms we consider for these experiments are our proposed improved greedy algorithms from chapter 3 (Gershgorin Disc GD, Correlation CCW, Frobenius FCW), Expectation-Maximization (EM) [15], Approximate Greedy (Path) [8], GPower algorithms (here we consider GPower0c, GPower0p, GPower1p ℓ_0 or ℓ_1 norm constrained or penalized version of PCA) [16, 18], Greedy and Partial Coordinate-Wise Algorithms (GCW, PCW) [20]. We omit the GPBB-ls algorithm because of its underwhelming results from the previous section. For these algorithms, we used the same implementation from 5.1 which also performs post-processing, except for PCW and GCW which are implemented with greedy eigenvalue computations instead of one-dimensional problems. Hence, on termination of an algorithm, the stationarity condition is imposed, and the dominant eigenpair of the sparse subspace, i.e., the eigenpair corresponding to the sparsity pattern constructed, is computed.

For some fixed sparsity level, we deploy the algorithms mentioned above. Then we set the best heuristic value as the variance constraint in our formulation. Then we apply PCA Sparsified. Throughout the reweighted optimization iterations, we consider X and its dominant eigenvector $v_1(X)$. Furthermore, $v_1(X)$ is truncated

[2] Our implementations of PCA Sparsified are available on: https://github.com/Fatih-S-AKTAS/PCA-Sparsified.

by the sparsity level, and the sparse eigenpair corresponding to the sparsity pattern discovered is computed. Finally, we return the best sparse eigenpair discovered, as in our experiments the final computed X does not always have the best solution to the SPCA problem. This behaviour is expected by the design of PCA Sparsified, as the objective is not maximizing the variance but implicitly doing so to construct small weighted ℓ_1 norm solutions that are also sparse.

5.2.1 Random Data

We generated random matrices $B \in \mathbb{R}^{m \times n}$ by sampling from i.i.d. standard Gaussian distribution ($B_{ij} \sim \mathcal{N}(0, 1)$). We solved problems having numbers of variables $n = 200, 400$ and for each sparsity level, we considered both underdetermined and overdetermined cases.

We omit all GPower results except for cardinality constrained versions because of their stability and performance issues for the case where the number of features is equal to 400.

As can be seen from Figures 5.7b and 5.6b, in overdetermined cases PCA Sparsified works well outperforming other local algorithms. Figure 5.6a shows that in the underdetermined scenario PCA Sparsified is still better although the margin is smaller. However, in Figure 5.7a sometimes even other algorithms were able to construct better sparse components than PCA Sparsified. This is mainly caused by the choice of stability parameter. In overdetermined cases, we were able to find a good choice of parameter that worked well in general. However, in underdetermined cases and in general in higher dimensions, a larger correction term is favored by PCA Sparsified due to our scaling scheme of the weights for the next iteration. In these experiments, we used a simple stability parameter value that depends on sparsity. In higher dimensions, our algorithm is more sensitive to the choice of stability parameter which may need to be adjusted, especially for data matrices $B \in \mathbb{R}^{m \times n}$ with $m << n$.

Behaviour of the greedy local algorithms seem to be the same as observed

in 5.1. In underdetermined cases and especially for the low sparsity levels very competitive with the other algorithms while not as competent in overdetermined cases or when the sparsity level is larger.

Algorithm\Sparsity	s = 80	s = 100	s = 120
Best Heuristic	617.56	666.875	710.2649
ADMM $\epsilon = (2s)^{-1}$	610.7	661.30	703.95
CGAL $\epsilon = (4s)^{-1}$	589.33	656.56	703.93
ADMM ϵ adjusted	**620.83**	670.95	**712.77**
CGAL ϵ adjusted	619.63	**671.03**	711.03

Table 5.2: Illustration of effect of choice of stability parameter for the underdetermined dataset with feature size $n = 400$

Table 5.2 presents the explained variance results obtained with larger stability parameters for the underdetermined data, which shows again the power of PCA Sparsified over other methods with a tuned stability parameter. These results are aligned with those in [45] in the sense that the choice of stability parameter ϵ affects the quality of the solution. Taking $\epsilon \to 0$ to make the log-sum penalty look more ℓ_0 like often results in getting stuck in a bad local minimum while setting ϵ too large diminishes the effect of reweighted optimization. In practice we found that choosing $\epsilon \in [1/(2s), 2/s]$ often works well. Finally, we set the number of reweighted optimization iterations to 20. We note that further gains are possible with a larger number of reweighted optimization iterations in the same test cases. Nevertheless, these variance gains come at a high computational cost. At each iteration of reweighted optimization, an SDP subproblem is solved. Moreover, each subproblem requires computationally challenging oracles like projection for ADMM and linear minimization oracle for CGAL. However, the computational cost does not depend on the target sparsity level, which is the main reason exact and convoluted combinatorial approaches like greedy forward or backward selection algorithms are computationally intractable.

5.2.2 Computation Times

In this section we fixed the sparsity level $s = 40$ and number of observations $m = 2000$, and considered data matrices of size $n = 100, 200, ..., 1000$. Our choice for the number of observations stems from the observations from the previous numerical results. That is, finding a suitable stability parameter that works well is easier for overdetermined systems than underdetermined ones. We report the variance of the components as well as the computation times of the algorithms. Figure 5.8 summarizes the results obtained.

From the CPU and Wall times of the algorithms we see that PCA Sparsified is much more expensive than other local algorithms. Thus we omitted the computations times in subsection 5.2.1, as it is uninformative. PCA Sparsified is not meant to be competitive with the afore-mentioned algorithms in terms of computation times but rather in its ability to construct the same variance level at a sparser level or higher variance at the same sparsity level.

In higher dimensions, using CGAL for the PCA Sparsified subproblems seems to fail to construct high quality solutions. We believe this can be attributed to its parameters. While ADMM is adaptive in its augmented Lagrangian parameter and thus more robust to the initial choice, CGAL parameters are fixed and may need to be adjusted for higher dimensions. It requires one parameter for the augmented Lagrangian and another for the smoothing, which we kept at the same value for all test instances.

5.2.3 Variance Throughout Outer Iterations

We have shown in subsection 4.3.2 that the first step of PCA Sparsified is equivalent to DSPCA up to optimal Lagrange multiplier. Thus, any sparse component recovered by DSPCA can be also equivalently recovered by PCA Sparsified in its first step. In that sense, PCA Sparsified's starting point is DSPCA, and therefore, it will always surpass (or at least equal) DSPCA in terms of variance of the

component. Hence, instead of comparing these two algorithms or other numerical algorithms developed to solve DSPCA, we show the impact of reweighted optimization in enhancing sparsity in the components.

In Figure 5.9 we give empirical evidence that justifies our sparse component extraction strategy described: namely, the sparsifying effect of reweighted optimization. Our interpretation is that PCA Sparsified can explain the same variance level with a smaller cardinality, thus adding additional variables to that subset according to their magnitude in the solution allows our algorithm to surpass others. Red star marks the maximum variance and iteration where it is discovered. On the same dataset, for sparsity level equal to 20 and 60, monotonic behaviour is observed in terms of variance for the sparsity constrained problem. On the other hand, for sparsity levels equal to 10 and 40, after the maximum variance discovery, further sparser solutions that satisfy the same variance constraint are found. That is the reason why the performance of the algorithm deteriorates when the eigenvector is truncated by the largest 10 or 40 elements in the subsequent iterations. Note that, for sparsity levels 10 and 20, with small adjustments in the parameters, PCA Sparsified can recover the same variance for $s = 10$ and a higher variance for $s = 20$.

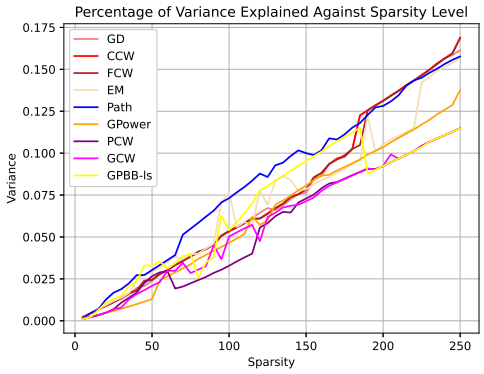
5.2.4 Results on Real Data

Finally, we show our numerical results on real datasets. We use Arrhythmia and FMA [82, 83] datasets that are available on UCI Machine Learning Repository.

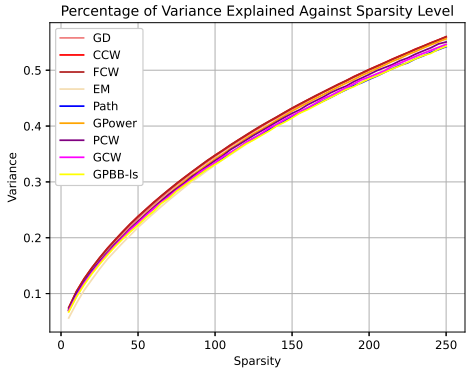
Arrhythmia Dataset: This dataset contains biological data comprised of 279 features and 452 observations. We drop 5 categorical variables and another 17 constant variables to get a correlation matrix of size 257 x 257.

FMA Dataset: This music dataset consists of 518 audio features extracted from 106,574 tracks. We include all of the variables to get a 518 x 518 correlation matrix.

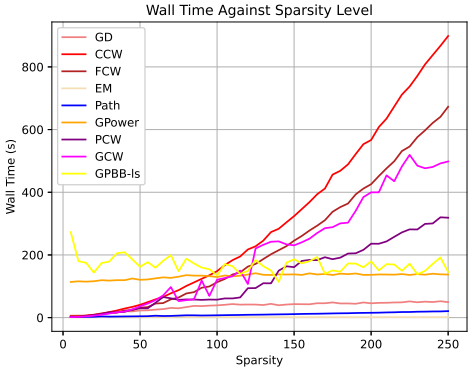
The figures demonstrate that PCA Sparsified is competitive with other methods on real datasets. In particular, for the FMA dataset, it seems to be much better for lower sparsity levels while for larger sparsity levels and for the most sparsity levels on the Arrhythmia dataset, local methods and PCA Sparsified appear to give similar results. However, we note that there were a few cases where PCA Sparsified recovered a variance that is smaller than what the best local method could recover with a relative difference of 10^{-4}, which is not observable from the figures. Nevertheless, with a search over a wider range of stability parameter values or introducing perturbations at variance level, larger variances may be obtained in those cases as well. In general, real data may be expected to possess some specific features which may be exploited for judicious parameter selection and fine-tuning, leading to better performance in specific applications. This is an application dependent issue that we leave for further research.



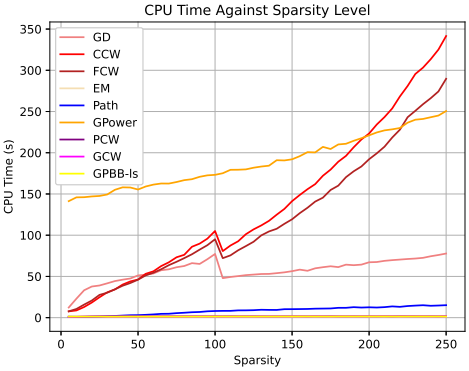
(a) Percentage of Explained Adjusted Variance of 5 Components for Changing Sparsity Levels on AT & T Faces Dataset



(b) Percentage of Explained Adjusted Variance of 10 Components for Changing Sparsity Levels on Random Dataset of Size 100×2000

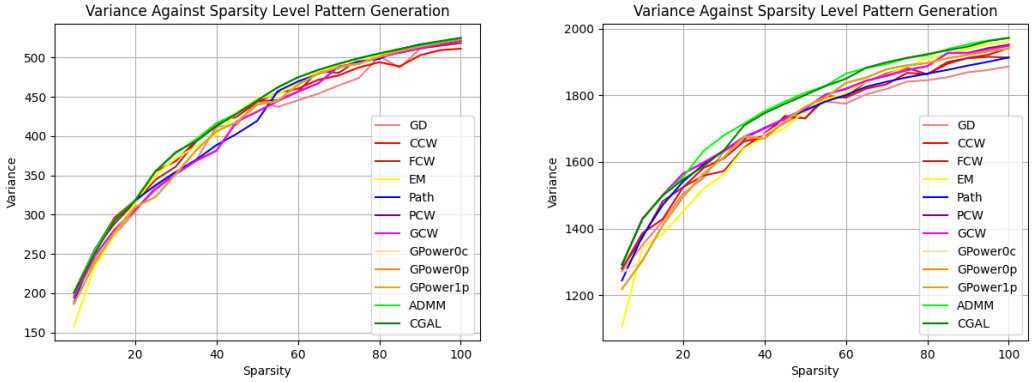


(c) Wall Time for Computing 5 Components for Changing Sparsity Levels on AT & T Faces Dataset



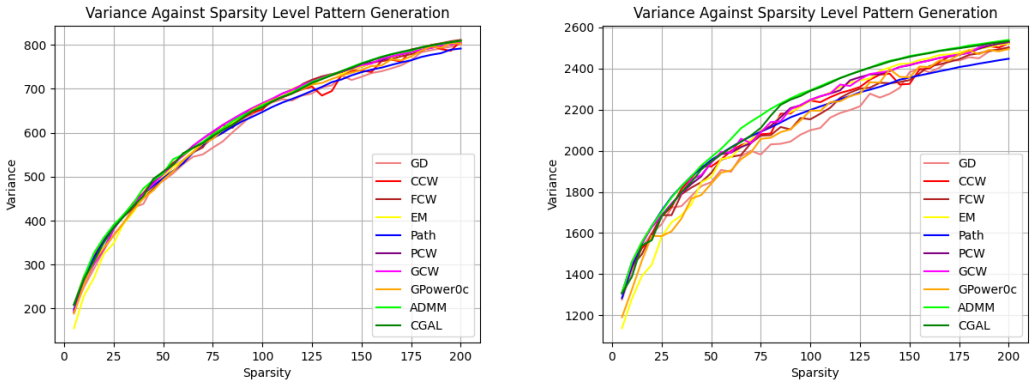
(d) Wall Time for Computing 5 Components for Changing Sparsity Levels on Random Dataset of Size 100×2000

Figure 5.5: Run Time and Variance Explained on AT & T and Arcene Datasets for the Multiple Components Problem



(a) Underdetermined data with $m = 100$ (b) Overdetermined data with $m = 1000$

Figure 5.6: Number of features in the data $n = 200$



(a) Underdetermined data with $m = 100$ (b) Overdetermined data with $m = 1000$

Figure 5.7: Number of features in the data $n = 400$

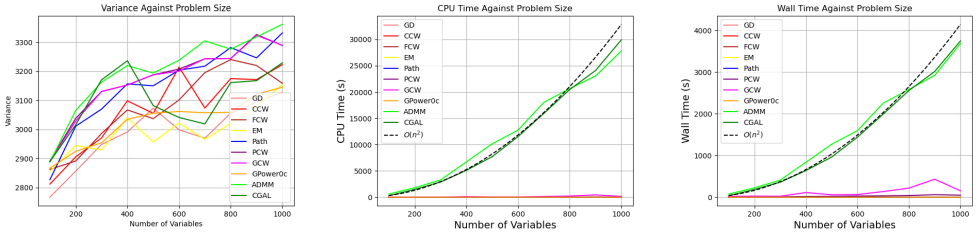
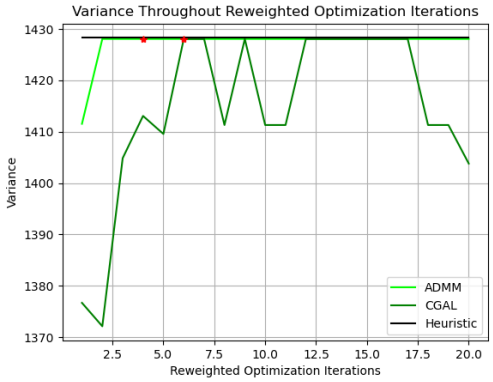
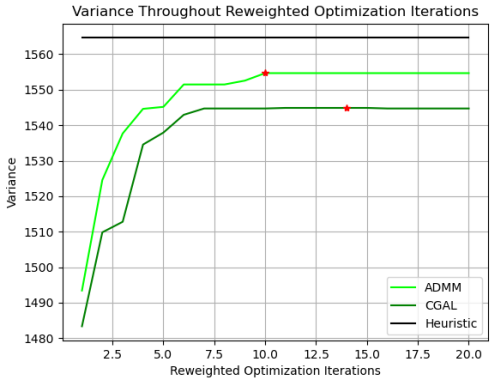


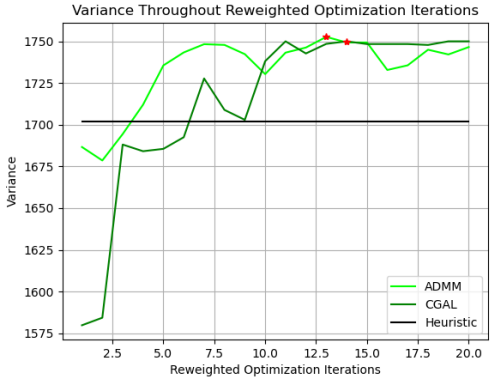
Figure 5.8: Variance and Computation times of the algorithms for varying dimension size. The empirical complexity of the algorithms seems to be $O(n^2)$.



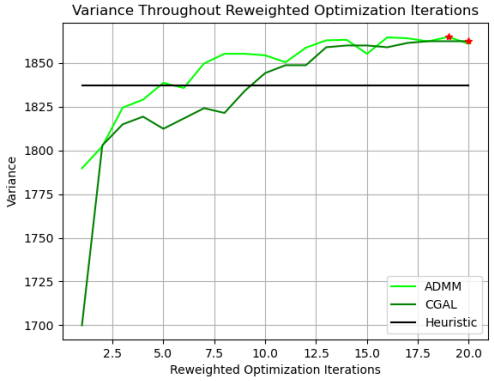
(a) Sparsity level = 10



(b) Sparsity level = 20

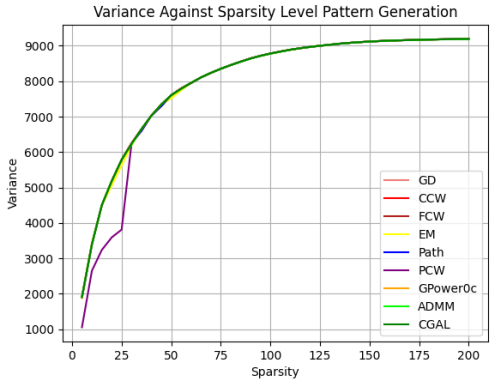


(c) Sparsity level = 40

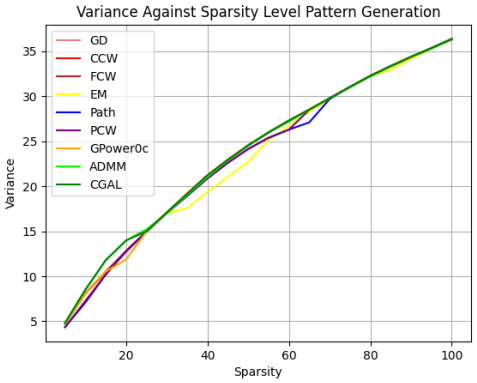


(d) Sparsity level = 60

Figure 5.9: Illustration of the sparsifying effect of reweighted optimization on overdetermined data with $n = 200$ for sparsity levels $10, 20, 40,$ and60



(a) Arrhythmia Dataset



(b) FMA Dataset

Figure 5.10: Results on Real Datasets

Chapter 6

Conclusion

We have proposed two different approaches to the SPCA Problem in this thesis. Our methods use different ideas and have different capabilities. While our first approach to the problem is simpler and not always guaranteed to produce very high-quality solutions compared to more complex approaches, it is computationally very cheap and works well in large problems. On the other hand, our second approach is analytically justified and works very well in small problems, it is computationally very demanding.

In our first approach, we studied various greedy algorithms to solve the Sparse PCA problem efficiently. We used eigenvalue approximation methods to select possible candidates for the solution. Building a foundation for our algorithms, we made a theoretical analysis by tying our work to recently proposed approximation methods. Furthermore, we proposed to strengthen our algorithms with a two-step approach if desired. We conducted detailed numerical tests to compare our algorithms with the well-known methods from the literature. On real and on synthetic data, especially for the covariance matrices generated by underdetermined systems, our algorithms yield more accurate solutions compared to their competitors while being computationally as viable as the other efficient methods.

In our second approach, we proposed a new inverted approach to the sparsity constrained variance maximization (SPCA) problem. We approximated the problem using an SDP model and a non-convex log-sum penalty. We presented an iterative reweighted ℓ_1 norm minimization algorithm (PCA Sparsified) to solve this problem locally. We showed that every limit point of a sequence generated by PCA Sparsified is a stationary point of the non-convex surrogate function that approximates the ℓ_0 norm, the distance between successive iterates tends to zero, and the objective function value of the surrogate function monotonically decreases to a limit. We also developed two first-order algorithms with different computational requirements and error estimates and discussed their efficient implementations. In addition, we pointed out the connection of PCA Sparsified to previously proposed approaches and its adaptation to multiple components problem. Moreover, we presented a pre-processing method to shrink the problem size that can also be used with other similar approaches such as DSPCA. Our numerical results on random datasets show that PCA Sparsified outperforms other methods in terms of variance explained. Further numerical results on real data indicate that PCA Sparsified is competitive with other methods in explaining variance if not always superior.

Data Availability.

The data used in these studies are publicly available from the references indicated for each data set.

Bibliography

[1] O. Alter, P. O. Brown, and D. Botstein, "Singular value decomposition for genome-wide expression data processing and modeling," *Proceedings of the National Academy of Sciences*, vol. 97, no. 18, pp. 10101–10106, 2000.

[2] A. L. Price, N. J. Patterson, R. M. Plenge, M. E. Weinblatt, N. A. Shadick, and D. Reich, "Principal components analysis corrects for stratification in genome-wide association studies," *Nature Genetics*, vol. 38, no. 8, pp. 904–909, 2006.

[3] M. Ringnér, "What is principal component analysis?," *Nature Biotechnology*, vol. 26, no. 3, pp. 303–304, 2008.

[4] J. Zhang, Y. Yan, and M. Lades, "Face recognition: eigenface, elastic matching, and neural nets," *Proceedings of the IEEE*, vol. 85, no. 9, pp. 1423–1435, 1997.

[5] L. J. Hargrove, G. Li, K. B. Englehart, and B. S. Hudgins, "Principal components analysis preprocessing for improved classification accuracies in pattern-recognition-based myoelectric control," *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 5, pp. 1407–1414, 2008.

[6] B. Moghaddam, Y. Weiss, and S. Avidan, "Generalized spectral bounds for sparse LDA," in *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, (New York, NY, USA), p. 641–648, Association for Computing Machinery, 2006.

[7] A. d'Aspremont, "Identifying small mean-reverting portfolios," *Quantitative Finance*, vol. 11, no. 3, pp. 351–364, 2011.

[8] A. d'Aspremont, F. Bach, and L. E. Ghaoui, "Optimal solutions for sparse principal component analysis," *Journal of Machine Learning Research*, vol. 9, p. 1269–1294, June 2008.

[9] R. Luss and A. d'Aspremont, "Clustering and feature selection using sparse principal component analysis," *Optimization and Engineering*, vol. 11, no. 1, pp. 145–157, 2010.

[10] Y. Zhang and L. Ghaoui, "Large-scale sparse principal component analysis with application to text data," in *Advances in Neural Information Processing Systems* (J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, eds.), vol. 24, Curran Associates, Inc., 2011.

[11] J. Cadima and I. Jolliffe, "Loading and correlations in the interpretation of principle components," *Journal of Applied Statistics*, vol. 22, no. 2, pp. 203–214, 1995.

[12] 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.

[13] H. Zou, T. Hastie, and R. Tibshirani, "Sparse principal component analysis," *Journal of Computational and Graphical Statistics*, vol. 15, pp. 262–286, 2006.

[14] W. W. Hager, D. T. Phan, and J. Zhu, "Projection algorithms for nonconvex minimization with application to sparse principal component analysis," *Journal of Global Optimization*, vol. 65, no. 4, pp. 657–676, 2016.

[15] C. D. Sigg and J. M. Buhmann, "Expectation-maximization for sparse and non-negative PCA," *Proceedings of the 25th International Conference on Machine Learning - ICML 08*, 2008.

[16] M. Journée, Y. Nesterov, P. Richtárik, and R. Sepulchre, "Generalized power method for sparse principal component analysis," *Journal of Machine Learning Research*, vol. 11, p. 517–553, Mar. 2010.

[17] R. Luss and M. Teboulle, "Conditional gradient algorithms for rank-one matrix approximations with a sparsity constraint," *SIAM Review*, vol. 55, pp. 65–98, 2013.

[18] P. Richtárik, M. Jahani, S. D. Ahipaşaoğlu, and M. Takáč, "Alternating maximization: unifying framework for 8 sparse PCA formulations and efficient parallel codes," *Optimization and Engineering*, vol. 22, pp. 1493–1519, Sept. 2020.

[19] B. Moghaddam, Y. Weiss, and S. Avidan, "Spectral bounds for sparse PCA: Exact and greedy algorithms," in *Advances in Neural Information Processing Systems* (Y. Weiss, B. Schölkopf, and J. Platt, eds.), vol. 18, MIT Press, 2006.

[20] A. Beck and Y. Vaisbourd, "The sparse principal component analysis problem: Optimality conditions and algorithms," *Journal of Optimization Theory and Applications*, vol. 170, pp. 119–143, Apr. 2016.

[21] A. d'Aspremont, L. El Ghaoui, M. Jordan, and G. Lanckriet, "A direct formulation of sparse PCA using semidefinite programming," *SIAM Review*, vol. 49, no. 3, 2007.

[22] R. Luss and M. Teboulle, "Convex approximations to sparse PCA via lagrangian duality," *Operations Research Letters*, vol. 39, no. 1, p. 57–61, 2011.

[23] Z. Lu and Y. Zhang, "An augmented lagrangian approach for sparse principal component analysis," *Mathematical Programming*, vol. 135, no. 1, pp. 149–193, 2012.

[24] V. Q. Vu, J. Cho, J. Lei, and K. Rohe, "Fantope projection and selection: A near-optimal convex relaxation of sparse PCA," *Advances in Neural Information Processing Systems*, vol. 26, 2013.

[25] A. A. Amini and M. J. Wainwright, "High-dimensional analysis of semidefinite relaxations for sparse principal components," in *2008 IEEE International Symposium on Information Theory*, pp. 2454–2458, IEEE, 2008.

[26] J. Lei and V. Q. Vu, "Sparsistency and agnostic inference in sparse PCA," *Annals of Statistics*, vol. 43, no. 1, pp. 299–322, 2015.

[27] V. Q. Vu and J. Lei, "Minimax sparse principal subspace estimation in high dimensions," *Annals of Statistics*, vol. 41, no. 6, pp. 2905–2947, 2013.

[28] A. Farcomeni, "An exact approach to sparse principal component analysis," *Computational Statistics*, vol. 24, pp. 583–604, 12 2009.

[29] L. Berk and D. Bertsimas, "Certifiably optimal sparse principal component analysis," *Mathematical Programming Computation*, vol. 11, no. 3, p. 381–420, 2019.

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

[31] Y. Li and W. Xie, "Exact and approximation algorithms for sparse pca," *arXiv preprint arXiv:2008.12438*, 2020.

[32] S. S. Dey, R. Mazumder, and G. Wang, "A convex integer programming approach for optimal sparse pca," *arXiv preprint arXiv:1810.09062*, 2018.

[33] S. O. Chan, D. Papailliopoulos, and A. Rubinstein, "On the approximability of sparse pca," in *29th Annual Conference on Learning Theory* (V. Feldman, A. Rakhlin, and O. Shamir, eds.), vol. 49 of *Proceedings of Machine Learning Research*, (Columbia University, New York, New York, USA), pp. 623–646, PMLR, 23–26 Jun 2016.

[34] Y. Li and W. Xie, "Beyond symmetry: Best submatrix selection for the sparse truncated svd," 2021.

[35] S. Gerschgorin, "Uber die abgrenzung der eigenwerte einer matrix," *Izvestija Akademii Nauk SSSR, Serija Matematika*, vol. 7, no. 3, pp. 749–754, 1931.

[36] A. Brauer *et al.*, "Limits for the characteristic roots of a matrix ii," *Duke Math. J*, vol. 14, no. 1, pp. 21–26, 1947.

[37] R. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations* (R. Miller and J. Thatcher, eds.), pp. 85–103, Plenum Press, 1972.

[38] G. H. Golub and C. F. van Loan, *Matrix Computations*. JHU Press, fourth ed., 2013.

[39] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

[40] M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan, "Time bounds for selection," *Journal of Computer and System Sciences*, vol. 7, p. 448–461, Aug. 1973.

[41] D. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1999.

[42] X.-T. Yuan and T. Zhang, "Truncated power method for sparse eigenvalue problems," *Journal of Machine Learning Research*, vol. 14, p. 899–925, Apr. 2013.

[43] I. Jolliffe, *Principal Component Analysis*. Springer Verlag, 1986.

[44] L. Mackey, "Deflation methods for sparse PCA," in *Advances in Neural Information Processing Systems* (D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, eds.), vol. 21, Curran Associates, Inc., 2009.

[45] E. J. Candes, M. B. Wakin, and S. P. Boyd, "Enhancing sparsity by reweighted ℓ_1 minimization," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5, pp. 877–905, 2008.

[46] K. Mohan and M. Fazel, "Iterative reweighted algorithms for matrix rank minimization," *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 3441–3473, 2012.

[47] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[48] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM Review*, vol. 52, no. 3, pp. 471–501, 2010.

[49] R. Tibshirani, "Regression shrinkage and selection via the LASSO," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[50] F. Alizadeh, "Interior point methods in semidefinite programming with applications to combinatorial optimization," *SIAM Journal on Optimization*, vol. 5, no. 1, pp. 13–51, 1995.

[51] L. Lovász and A. Schrijver, "Cones of matrices and set-functions and 0–1 optimization," *SIAM Journal on Optimization*, vol. 1, 01 1990.

[52] R. T. Rockafellar, *Convex Analysis*. Princeton University Press, 2015.

[53] M. Fazel, *Matrix rank minimization with applications*. PhD thesis, Stanford University, 2002.

[54] K. Lange, *Optimization*. Springer Texts in Statistics, Springer, New York, 2013.

[55] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, p. 1–122, Jan. 2011.

[56] A. Yurtsever, O. Fercoq, and V. Cevher, "A conditional-gradient-based augmented lagrangian framework," in *Proc. 36th Int. Conf. Machine Learning (ICML)*, 2019.

[57] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in Machine Learning*, vol. 1, p. 127–239, jan 2014.

[58] M. Hestenes, "Multiplier and gradient methods," *Journal of Optimization Theory and Applications*, vol. 4, pp. 303–320, 1969.

[59] W. W. Hager, "Updating the inverse of a matrix," *SIAM Review*, vol. 31, no. 2, pp. 221–239, 1989.

[60] C. Chen, B. He, Y. Ye, and X. Yuan, "The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent," *Mathematical Programming*, vol. 155, p. 57–79, jan 2016.

[61] Y. Nesterov, "Complexity bounds for primal-dual methods minimizing the model of objective function," *Mathematical Programming*, vol. 171, no. 1, pp. 311–330, 2018.

[62] Y. Nesterov, "Smooth minimization of non-smooth functions," *Mathematical Programming*, vol. 103, no. 1, pp. 127–152, 2005.

[63] P. Lions and B. Mercier, "Splitting algorithms for the sum of two nonlinear operators," *SIAM Journal on Numerical Analysis*, vol. 16, no. 6, pp. 964–979, 1979.

[64] B. He and X. Yuan, "On the $O(1/n)$ convergence rate of the douglas–rachford alternating direction method," *SIAM Journal on Numerical Analysis*, vol. 50, no. 2, pp. 700–709, 2012.

[65] A. Yurtsever, J. A. Tropp, O. Fercoq, M. Udell, and V. Cevher, "Scalable semidefinite programming," *SIAM Journal on Mathematics of Data Science*, vol. 3, no. 1, pp. 171–200, 2021.

[66] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Review*, vol. 53, no. 2, pp. 217–288, 2011.

[67] J. Kuczyński and H. Woźniakowski, "Estimating the largest eigenvalue by the power and lanczos algorithms with a random start," *SIAM Journal on Matrix Analysis and Applications*, vol. 13, no. 4, pp. 1094–1122, 1992.

[68] G. H. Golub and C. F. Van Loan, *Matrix Computations*. The Johns Hopkins University Press, third ed., 1996.

[69] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[70] M. Jaggi, "Revisiting Frank-Wolfe: Projection-free sparse convex optimization," in *Proceedings of the 30th International Conference on Machine Learning* (S. Dasgupta and D. McAllester, eds.), vol. 28 of *Proceedings of Machine Learning Research*, (Atlanta, Georgia, USA), pp. 427–435, PMLR, 17–19 Jun 2013.

[71] S. Lacoste-Julien, "Convergence rate of frank-wolfe for non-convex objectives," 2016.

[72] V. Leplat, Y. Nesterov, N. Gillis, and F. Glineur, "Conic-optimization based algorithms for nonnegative matrix factorization," 2023.

[73] M. Sion, "On general minimax theorems.," *Pacific Journal of Mathematics*, vol. 8, no. 1, pp. 171–176, 1958.

[74] A. Beck, *First-Order Methods in Optimization*. SIAM, 2017.

[75] Y. Nesterov, *Lectures on Convex Optimization*. Springer Publishing Company, Incorporated, 2nd ed., 2018.

[76] Y. Ye, "Conic linear programming." Available at https://web.stanford.edu/class/msande314/, December 2004. Revised, October 2017.

[77] G. Still and M. Streng, "Optimality conditions in smooth nonlinear programming," *Journal of optimization theory and applications*, vol. 90, pp. 483–515, 1996.

[78] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. Software, environments, tools, SIAM, 1998.

[79] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*. Philadelphia, PA: Society for Industrial and Applied Mathematics, third ed., 1999.

[80] F. Chen and K. Rohe, "A new basis for sparse PCA," 2020.

[81] J. N. R. Jeffers, "Two case studies in the application of principal component analysis," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 16, no. 3, pp. 225–236, 1967.

[82] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "FMA: A dataset for music analysis," in *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[83] D. Dua and C. Graff, "UCI machine learning repository," 2017.

Appendix A

Sufficient Condition for Convergence of ADMM

We begin by rewriting model (4.15) using $x_1 = \text{vec}\,(X_1)$, $x_2 = \text{vec}\,(X_2)$, $x_3 = \text{vec}\,(X_3)$, $a = \text{vec}\,(A)$, $v = \text{vec}\,(I_n)$, and $D_W = \text{diag}(\text{vec}\,(W))$, where $\text{diag}(y)$ operator constructs a diagonal matrix of appropriate size from vector y.

$$\begin{aligned} \min_{x_1,x_2,x_3} \quad & ||D_W x_1||_1 + \mathcal{I}_{\mathbf{S}^n_+}(\text{mat}\,(x_3)) \\ \text{st.} \quad & \begin{bmatrix} I_{n^2} & -I_{n^2} & 0 \\ 0 & -I_{n^2} & I_{n^2} \\ 0 & a & 0 \\ 0 & v & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ b \\ 1 \end{bmatrix} \end{aligned} \tag{A.1}$$

Now we deploy one of the sufficient conditions proved in [60] for convergence of ADMM with 3-blocks; $A_1^T A_3 = 0$. From Model (A.1) it is clear that this problem satisfies this condition and therefore ADMM is guaranteed to converge to a global optimum. Finally, remark that in [60] any order of the optimization is allowed as long as the orthogonality property $A_i^T A_j = 0$ among at least one pair is satisfied.

Appendix B

ADMM Residual Calculations

Recall that the necessary and sufficient optimality conditions for the ADMM problem in 4.15 are primal feasibility

$$\begin{aligned} &X_1 - X_2 = 0 \\ &-X_2 + X_3 = 0 \\ &\mathrm{tr}(AX_2) - b = 0 \\ &\mathrm{tr}(X_2) - 1 = 0 \\ &X_3 \in \mathbf{S}_+^n \end{aligned} \tag{B.1}$$

and dual feasibility

$$\begin{aligned} &0 \in \partial ||W \circ X_1||_1 + U_1 \\ &0 = y_1 A + y_2 I - U_1 - U_2 \\ &0 \in \partial \mathcal{I}_{\mathbf{S}_+^n}(X_3) + U_2 \end{aligned} \tag{B.2}$$

where the ∂ symbol denotes the subdifferential set [74, 75]. However, in the ADMM algorithm, these conditions are not enforced, i.e. ADMM does not directly compute a solution to these systems of equations, yet similar conditions are enforced via sequential minimization over X_i's and dual corrections over y_i's and U_i's.

Consider the minimization over X_3 at iteration $(k+1)$.

$$\begin{aligned} X_3^{(k+1)} &= \arg\min_{X_3} \mathcal{L}_\rho(X_1^{(k+1)}, X_2^{(k+1)}, X_3, y^{(k)}, U^{(k)}) \\ \implies \quad & 0 \in \partial \mathcal{I}_{\mathbf{S}_+^n}(X_3^{(k+1)}) + U_2^{(k)} + \rho(X_3^{(k+1)} - X_2^{(k+1)}) \\ \implies \quad & 0 \in \partial \mathcal{I}_{\mathbf{S}_+^n}(X_3^{(k+1)}) + U_2^{(k+1)} \end{aligned} \tag{B.3}$$

Thus at the end of iteration $(k+1)$, $X_3^{(k+1)}$ and $U_2^{(k+1)}$ satisfy the dual feasibility criterion for X_3 in the original problem. Therefore, it remains to check the dual feasibility with respect to X_1 and X_2 and primal feasibility. Checking the optimality conditions for X_2 minimization at iteration $(k+1)$ gives

$$\begin{aligned} X_2^{(k+1)} &= \arg\min_{X_2} \mathcal{L}_\rho(X_1^{(k+1)}, X_2, X_3^{(k)}, y^{(k)}, U^{(k)}) \\ \implies \quad & 0 = y_1^{(k)} A + y_2^{(k)} I - U_1^{(k)} - U_2^{(k)} + \rho A(\mathrm{tr}(AX_2^{(k+1)}) - b) + \\ & \rho I(\mathrm{tr}(X_2^{(k+1)}) - 1) + \rho(X_2^{(k+1)} - X_1^{(k+1)}) + \rho(X_2^{(k+1)} - X_3^{(k)}) \\ & 0 = y_1^{(k+1)} A + y_2^{(k+1)} I - U_1^{(k+1)} - U_2^{(k)} + \rho(X_2^{(k+1)} - X_3^{(k)}) \\ & 0 = y_1^{(k+1)} A + y_2^{(k+1)} I - U_1^{(k+1)} - U_2^{(k)} + \rho(X_2^{(k+1)} - X_3^{(k)}) + \rho X_3^{(k+1)} - \rho X_3^{(k+1)} \\ & 0 = y_1^{(k+1)} A + y_2^{(k+1)} I - U_1^{(k+1)} - U_2^{(k+1)} + \rho X_3^{(k+1)} - \rho X_3^{(k)} \\ & \rho(X_3^{(k)} - X_3^{(k+1)}) = y_1^{(k+1)} A + y_2^{(k+1)} I - U_1^{(k+1)} - U_2^{(k+1)} \end{aligned} \tag{B.4}$$

Thus we can use the term on the left-hand side as a dual residual regarding the optimality condition with respect to X_2. Similarly calculating the dual residual with respect to X_1 gives;

$$\begin{aligned} X_1^{(k+1)} &= \arg\min_{X_1} \mathcal{L}_\rho(X_1, X_2^{(k)}, X_3^{(k)}, y^{(k)}, U^{(k)}) \\ \implies \quad & 0 \in \partial ||W \circ X_1^{(k+1)}||_1 + U_1^{(k)} + \rho(X_1^{(k+1)} - X_2^{(k)}) \\ & 0 \in \partial ||W \circ X_1^{(k+1)}||_1 + U_1^{(k)} + \rho(X_1^{(k+1)} - X_2^{(k)}) + \rho X_2^{(k+1)} - \rho X_2^{(k+1)} \\ & 0 \in \partial ||W \circ X_1^{(k+1)}||_1 + U_1^{(k+1)} + \rho X_2^{(k+1)} - \rho X_2^{(k)} \\ & \rho(X_2^{(k)} - X_2^{(k+1)}) \in \partial ||W \circ X^{(k+1)}||_1 + U_1^{(k+1)} \end{aligned} \tag{B.5}$$

Similar to X_2 dual residual, we can see the term $\rho(X_2^{(k)} - X_2^{(k+1)})$ as dual residual with respect to X_1 optimality condition. Finally, we can use B.1 directly for the primal residual at the end of a full ADMM iteration with the most recent values of primal and dual variable values.

Appendix C

CGAL Smoothing Details

In order to obtain a smooth approximation to the ℓ_1 norm, we first rewrite it in terms of its dual norm characterization

$$h(X) = \|W \circ X\|_1 = \max_{||U||_\infty \leq 1} \langle U, W \circ X \rangle. \tag{C.1}$$

A smooth approximation is obtained by adding a strongly convex penalty term to the maximization problem with some smoothing parameter $\beta > 0$

$$h_\beta(X) = \|W \circ X\|_{H_\beta} = \max_{||U||_\infty \leq 1} \langle U, W \circ X \rangle - \beta d(U). \tag{C.2}$$

Consider the following strongly convex functions for $d(U)$: weighted Frobenius norm

$$d(U) = \tfrac{1}{2}||W^{\circ \frac{1}{2}} \circ U||_F^2 = \tfrac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} u_{ij}^2, \tag{C.3}$$

where W_{ij}'s are the same weights as in $||W \circ X||_1$. This is a strongly convex function with respect to the Frobenius norm with convexity parameter $\sigma = \min_{i,j} \{w_{ij}\}$. The center of the strongly convex function U_0 is equal to $\mathbf{0}$, which is defined as:

$$U_0 = \arg\min_{||U||_\infty \leq 1} \quad d(U), \tag{C.4}$$

and center satisfies $d(U_0) = 0$. Furthermore, the function satisfies the following envelop property

$$h_\beta(X) \leq h(X) \leq h_\beta(X) + \beta D, \tag{C.5}$$

where D is defined as:

$$D = \max_{||U||_\infty \leq 1} d(U) = \tfrac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} |w_{ij}|, \tag{C.6}$$

which determines the sharpness of the envelop property of smoothing. Finally, we compute the norm of the linear operator W which is defined as

$$\begin{aligned} ||W|| &= \max_{U,X} \langle U, W \circ X \rangle : ||U||_F = 1, ||X||_F = 1 \\ &= \max_{X} ||W \circ X||_F : ||X||_F = 1 \\ &= \max_{i,j} \{|W_{ij}|\}. \end{aligned} \tag{C.7}$$

We replace $h(X)$ with its smooth approximation $h_\beta(X)$. The function $h_\beta(X)$ is continuously differentiable with the gradient given as

$$\nabla h_\beta(X) = W \circ U_\beta(X), \tag{C.8}$$

where $U_\beta(X)$ is an optimal solution of problem (C.2). Moreover, $h_\beta(X)$ is Lipschitz continuous with constant

$$L_\beta = \frac{1}{\beta\sigma} ||W||^2, \tag{C.9}$$

where σ is the strong convexity parameter of the penalty function. For more details and proofs we refer to [62].

Appendix D

Proof of convergence in distance between consecutive X

We first give convergence in X for the good case where U attains the stationarity, i.e., $U^{(t+1)} = U^{(t)}\ \exists\, t \in \mathbb{N}$.

Lemma 1. *If for some $t \in \mathbb{N}$ we have $U^{(t+1)} = U^{(t)}$, then we must have $X^{(t+1)} = X^{(t)}$.*

Proof. Assume to the contrary, we have $U^{(t+1)} = U^{(t)}$ but $X^{(t+1)} \neq X^{(t)}$. Then consider $Y = 0.5(X^{t+1)} + X^{(t)})$. Since the feasible region is convex, Y is a feasible solution for the problem. Since we have $U^{(t+1)} = U^{(t)}$, we have $|X_{ij}^{(t+1)}| = |X_{ij}^{(t)}|\,\forall\,(i,j)$. Then by our assumption; $\exists\,(i,j)$ s.t $X_{ij}^{(t+1)} = -X_{ij}^{(t)}$, where $|X_{ij}^{(t)}| >$

0. Then, the objective value of Y is;

$$
\begin{aligned}
||W \circ Y||_1 &= \sum_{(i,j):X_{ij}^{(t+1)}=X_{ij}^{(t)}} \sum w_{ij}|y_{ij}| + \sum_{(i,j):X_{ij}^{(t+1)}=-X_{ij}^{(t)}} \sum w_{ij}|y_{ij}| \\
&= \sum_{(i,j):X_{ij}^{(t+1)}=X_{ij}^{(t)}} \sum 0.5w_{ij}|x_{ij}^{(t+1)} + x_{ij}^{(t)}| + \sum_{(i,j):X_{ij}^{(t+1)}=-X_{ij}^{(t)}} \sum 0.5w_{ij}|x_{ij}^{(t+1)} + x_{ij}^{(t)}| \\
&= \sum_{(i,j):X_{ij}^{(t+1)}=X_{ij}^{(t)}} \sum w_{ij}|x_{ij}^{(t+1)}| + \sum_{(i,j):X_{ij}^{(t+1)}=-X_{ij}^{[}(t)} \sum w_{ij}|x_{ij}^{(t+1)}| - w_{ij}|x_{ij}^{(t+1)}| \\
&= ||W \circ X^{(t+1)}||_1 - \sum_{(i,j):X_{ij}^{(t+1)}=-X_{ij}^{(t)}} \sum w_{ij}|x_{ij}^{(t+1)}| \\
&< ||W \circ X^{(t+1)}||_1,
\end{aligned}
\tag{D.1}
$$

where the fifth line follows from existence of (i, j) pair for nonzero $X_{ij}^{(t+1)}$. This contradicts the optimality of $X^{(t+1)}$ at iteration $(t+1)$. Therefore, if stationarity is attained for U, it will be attained in X. □

Proposition 3. *The distance between consecutive iterates of X converges even if U is always nearly stationary but stationarity is never attained, i.e., the distance between consecutive iterates of X always converges.*

Proof. Using previous convergence result obtained in section 4.5, for large t we have

$$||W \circ U^{(t+1)} - W \circ U^{(t)}||_1 < \varepsilon \tag{D.2}$$

for any given $\varepsilon > 0$, since any norm is equivalent in finite-dimensional spaces. This could also be seen as a convergence in objective as a consequence of convergence in the Frobenius norm. Now since $X^{(t+1)}$ is a minimizer at iteration $(t + 1)$ we

have

$$
\begin{aligned}
&||W \circ X^{(t)} + W \circ X^{(t+1)}||_1 - 2||W \circ X^{(t+1)}||_1 \geq 0 \\
&\sum_{(i,j)\in S}\sum w_{ij}(|x_{ij}^{(t)}| - |x_{ij}^{(t+1)}|) + \sum_{(i,j):S',|x_{ij}^{(t+1)}|>|x_{ij}^{(t)}|}\sum w_{ij}(-|x_{ij}^{(t)}| - |x_{ij}^{(t+1)}|) + \\
&\sum_{(i,j):S',|x_{ij}^{(t+1)}|<|x_{ij}^{(t)}|}\sum w_{ij}(|x_{ij}^{(t)}| - 3|x_{ij}^{(t+1)}|) \geq 0 \\
&||W \circ X^{(t)}||_1 - ||W \circ X^{(t+1)}||_1 \geq \sum_{(i,j):S',|x_{ij}^{(t+1)}|<|x_{ij}^{(t)}|}\sum w_{ij}|x_{ij}^{(t+1)}| + \\
&\sum_{(i,j):S',|x_{ij}^{(t+1)}|>|x_{ij}^{(t)}|}\sum w_{ij}|x_{ij}^{(t)}| \\
&\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}|u_{ij}^{(t)} - u_{ij}^{(t+1)}| \geq \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(u_{ij}^{(t)} - u_{ij}^{(t+1)}),
\end{aligned}
\tag{D.3}
$$

where we define $S = \{(i,j)|\,\text{sign}(x_{ij}^{(t+1)}) = \text{sign}(x_{ij}^{(t)})\}$ and $S' = \{(i,j)|\,\text{sign}(x_{ij}^{(t+1)}) \neq \text{sign}(x_{ij}^{(t)})\}$. Also, W and w_{ij} refer to weights at iteration $(t+1)$. (We implicitly use boundedness of weights, and appropriate choice of proximity for consecutive $U^{(k)}$'s if need be.) This shows that terms that have different signs tend to zero in the matrix with small elements. Since we have convergence in terms of the absolute value of elements, large magnitude elements with different signs in the other matrix also tend to zero, which concludes the proof. □

Appendix E

Pre-processing for DSPCA

Consider the first step of PCA Sparsified, which is equivalent to DSPCA for some penalty parameter $\rho > 0$. If a feature has a maximum off-diagonal element lower than penalty parameter ρ, and it does not have the largest diagonal element, i.e., it does not have the largest variance, then we can safely remove it from the problem.

To prove this property we shall argue by contradiction. To the contrary, assume that there is an optimal solution X to DSPCA with this property. Let i be the feature with this property. X and A can be partitioned with respect to support of X and the feature i as follows

$$X = \begin{bmatrix} Z & z_i & 0 \\ z_i^T & z_{ii} & 0 \\ 0 & 0 & 0 \end{bmatrix}, A = \begin{bmatrix} H & h_i & G^T \\ h_i^T & h_{ii} & g^T \\ G & g & J \end{bmatrix}. \tag{E.1}$$

Now consider the following matrix

$$Y = \begin{bmatrix} Z & 0 & 0 \\ 0 & z_{ii} & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{E.2}$$

where the matrix Y is equal to zero outside the support of X. Evaluating the objective function value (4.14) gives

$$\begin{aligned} \langle A, Y \rangle - \rho ||Y||_1 - \langle A, X \rangle + \quad & \rho ||X||_1 = 2(-z_i^T h_i + \rho ||z_i||_1) \\ \geq 2(-||z_i||_1 ||h_i||_\infty + \quad & \rho ||z_i||_1) \geq 0. \end{aligned} \tag{E.3}$$

The second line follows from Hölder inequality. If we have $||h_i||_\infty < \rho$ we have necessarily zero off-diagonal elements. In the case of equality, we can zero out the off-diagonal elements of feature z_i without loss in objective function value. Assume j is the feature with the largest variance. If we further assume that $h_{ii} \leq a_{jj}$, we can take the matrix Y as

$$Y = \begin{bmatrix} Z & 0 \\ 0 & 0 \end{bmatrix} + z_{ii} e_j e_j^T. \tag{E.4}$$

As before, if we have strict inequality, the feature is necessarily zero in the optimal solution, and in the case of equality, it can be erased without loss in the objective function value.

In PCA Sparsified, instead of the penalty parameter, we have a corresponding Lagrange multiplier derived in subsection 4.5.4. We can use the lower bound derived for the penalty parameter as a sufficient condition to eliminate some of the variables, to reduce the problem to a manageable size. We note that while this may hurt the overall process in the end, as a variable set to zero by DSPCA may be given a nonzero value in subsequent iterations by PCA Sparsified, yet more conservative pre-processing techniques may help with managing problem size.