Sparse Principal Component Analysis: Algorithms and Applications

Youwei Zhang

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2013-187
http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-187.html

December 1, 2013

Sparse Principal Component Analysis: Algorithms and Applications

by

Youwei Zhang

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering and Computer Sciences

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Laurent El Ghaoui, Chair
Professor Martin Wainwright
Professor Jasjeet Sekhon

Fall 2011

Sparse Principal Component Analysis: Algorithms and Applications

by

Youwei Zhang

Abstract

Sparse Principal Component Analysis: Algorithms and Applications

by

Youwei Zhang

Doctor of Philosophy in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Laurent El Ghaoui, Chair

The Sparse Principal Component Analysis (Sparse PCA) problem is a variant of the classical PCA problem. The goal of Sparse PCA is to achieve a trade-off between the explained variance along a normalized vector, and the number of non-zero components of that vector. Sparse PCA has a wide array of applications in machine learning and engineering. Unfortunately, this problem is also combinatorially hard and hence various sub-optimal algorithms and approximation formulations have been proposed to tackle it. In this dissertation, we first discuss convex relaxation techniques that efficiently produce good approximate solutions. We then describe several algorithms solving these relaxations as well as greedy algorithms that iteratively improve the solution quality.

The dissertation then focuses on solving the attractive formulation called DSPCA (a Direct formulation for Sparse PCA) for large-scale problems. Although Sparse PCA has apparent advantages compared to PCA, such as better interpretability, it is generally thought to be computationally much more expensive. We demonstrate the surprising fact that sparse PCA can be easier than PCA in practice, and that it can be

reliably applied to very large data sets. This comes from a rigorous feature elimination pre-processing result, coupled with the favorable fact that features in real-life data typically have rapidly decreasing variances, which allows for many features to be eliminated. We introduce a fast block coordinate ascent algorithm with much better computational complexity than the existing first-order ones. We provide experimental results obtained on text corpora involving millions of documents and hundreds of thousands of features.

Another focus of the dissertation is to illustrate the utility of Sparse PCA in various applications, ranging from senate voting and finance to text mining. In particular, we apply Sparse PCA to the analysis of text data, with online news as our focus. Our experimental results on various data sets illustrate how Sparse PCA can help organize a large corpus of text data in a user-interpretable way, providing an attractive alternative approach to topic models.

Contents

List of Figures

List of Tables

Acknowledgements

First, I want to thank my advisor, Prof. Laurent El Ghaoui, for guiding me through my PhD journey at U.C. Berkeley. When I first entered Berkeley and read the EECS graduate handbook saying that "a good advisor is your mentor, friend, confidant, supporter and problem solver", I was wondering whether there would ever exist such a perfect advisor. Today I have to be grateful for the fact that life does give me such an advisor. I am grateful for having an advisor as Laurent, who is constantly caring, encouraging and supportive. There is no need to mention how Laurent patiently teaches and guides me to do research. Everything I achieve here at Berkeley would not have been possible without the help and support from Laurent, literally. Even at this stage of my life, I must say my personality is still being able to be shaped by Laurent, as I have been trying my best to become a person as kind and polite as Laurent is to others.

I am also very grateful to Prof. Alexandre d'Aspremont. Alex coached me in writing a book chapter for Handbook on Semidefinite, Cone and Polynomial Optimization, which is the basis for the second chapter in this dissertation. Working with Alex has always been a very enjoyable and inspiring learning experience for me. I want to also thank Prof. Martin Wainwright and Prof. Jasjeet Sekhon for serving on my dissertation committee. I took one probability class and one statistics class with Martin, and he is one of the best instructors that I've ever had. His clear and insightful lectures lay a good foundation for my probability and statistical knowledge. As the outside member on my dissertation committee, Jas has always been very encouraging and approachable. I appreciate those kind and helpful discussions with him. I also thank Prof. Ken Goldberg for serving as the chair of my Qual committee and for his very useful comments and suggestions.

There are a long list of nice people at Berkeley that I would like to thank. Prof.

David Tse served as my temporary advisor when I first entered Berkeley and gave lots of initial support and help. Prof. Jean Walrand has been kindly helping me from the first semester till this last one at Berkeley. Jean taught me random processes, served as the chair of my Prelim committee and is one of my nice references for helping me land my full-time job after graduation. I've also been receiving help and care from Prof. Babak Ayazifar. Those early experience of working with Babak in teaching EE20 has always been one of my sweet memories. My graduate life at Berkeley has been made less stressful by the kind and knowledgable staff at EECS Grad Student Affairs such as Ruth Gjerde, Mary Byrnes, and Rebecca Miller, and at Berkeley International Office such as Rebecca Sablo. I also enjoy the interesting conversations and friendship with my teammates, Anh Pham, Tarek Rabbani, Vivian Viallon and Brian Gawalt during these years.

Finally, I want to thank my family for their understanding of and support for me pursuing my PhD in EECS at U.C. Berkeley.

Chapter 1

Introduction

Principal Component Analysis (PCA) (see e.g. Jolliffe [18] and Anderson [3]) is widely used for data analysis, visualization and dimension reduction. One way of performing PCA is to first calculate the sample covariance

$$\Sigma = \frac{1}{m} \sum_{i=1}^{m} x_i {x_i}^T$$

where $x_i \in \mathcal{R}^n, i = 1, ..., m$ are centered date samples. Then a set of eigenvectors $v_1, ..., v_k$ with top $k (k << n)$ eigenvalues are computed using eigenvalue decomposition and finally each data x is projected onto the set of eigenvectors to obtain a representation in $\mathcal{R}^k$.

An alternative way to obtain $v_1, ..., v_k$ is via singular value decomposition: first form a $m \times n$ data matrix X, where i^{th} row of X is ${x_i}^T$,

$$X = \begin{bmatrix} {x_1}^T \\ ... \\ {x_m}^T \end{bmatrix}$$

Then perform SVD: $X = UDV^T$, where $V = [v_1, ..., v_k, ..., v_n]$, $U = [u_1, ..., u_m]$ are both unitary.

Essentially, PCA finds linear combinations of the variables called *principal components* or factors, corresponding to orthogonal directions maximizing variance in the data. In this classical procedure, two issues in particular motivate Sparse PCA. First, each of the principal components $v_1, ..., v_k$ are typically a linear combination of all original variables. That is, most of coefficients (or loadings) in the principal components are non-zero. This means that while PCA facilitates model interpretation and visualization by concentrating the information in a few factors, the factors themselves are still constructed using all variables, hence are often hard to interpret. For example, with stock return data, we might expect to see a subset of stocks with some certain commonalities to be the driving force for the market volatility. Fig 1.1 gives an example of using PCA to visualize the S&P500 returns over one year. The top 2 principal components can explain 87% of the total variance. However, it is not quite clear about what the two principal components represent. Later in Chapter 4, we will present the interesting findings that Sparse PCA can reveal. Second, for some contemporary applications, the number of variables n can be comparable to or even larger than the number of available data points m. In that case, the sample estimator Σ is not a reliable one. Without further structure information, there is little hope to perform high dimensional inference with limited data. Sparsity in principal components could be a reasonable assumption in certain applications.

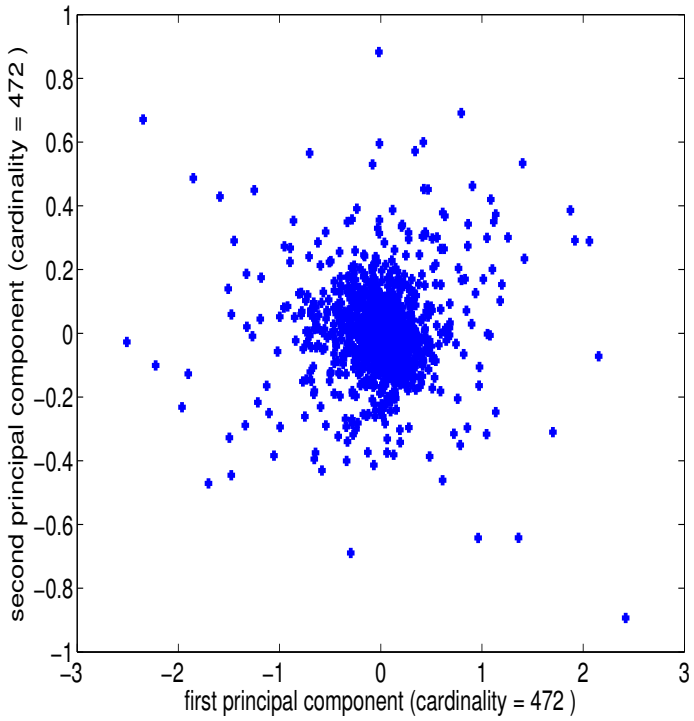

Figure 1.1: S&P500 daily returns projected onto the top 2 principal components.

The sparse principal component analysis (Sparse PCA) problem is a variant of the classical PCA problem, which accomplishes a trade-off between the explained variance along a normalized vector, and the number of non-zero components of that vector. In order to obtain sparse principal components, a simple thresholding is frequently used in practice: first use PCA to obtain principal components, and then set those loadings with absolute values less than some threshold to be zero. This method is quite ad hoc and can be misleading especially when there are only a limited number of data samples are available. Better approaches directly incorporate sparsity constraint into problem formulations. We discuss two such formulations in this dissertation.

The first formulation starts with the observation that the principal component

can be obtained via a variational representation:

$$\max \ z^T \Sigma z$$

$$\text{subject to } \|z\|_2 = 1$$

Where $z \in \mathbf{R}^n$, and $\Sigma \in \mathbf{S}_n$ is the (symmetric positive semi-definite) sample covariance matrix.The objective can be interpreted as the variance "explained by" (contained in) the direction z. Precisely, the ratio $\frac{z^T \Sigma z}{\mathbf{Tr}\, \Sigma}$ quantifies the amount of variance contained in the direction z v.s. the total variance.

By either adding a cardinality constraint $\mathbf{Card}(z) \leq k$ to the constraint set or adding a penalizing term $-\rho \mathbf{Card}(z)$ to the objective, we can obtain a non-convex optimization problem encouraging a sparse solution. Here ρ is a parameter controlling sparsity, and $\mathbf{Card}(z)$ denotes the cardinality (or ℓ_0 norm) of z, i.e. the number of non zero coefficients of z. Then relaxation techniques can be used to further convert the non-convex problem to a convex one, which will be discussed in Chapter 2. In this dissertation, we will focus on the version with the penalized objective as written below:

$$\max_{\|z\|_2=1} z^T \Sigma z - \rho\, \mathbf{Card}(z) \tag{1.1}$$

Another formulation comes from the viewpoint of low rank approximation of matrices. Given a centered data matrix $X \in \mathbf{R}^{m \times n}$ (where n is number of features and m is the number of samples),the leading principal component z can also be obtained by solving the following optimization problem

$$\min_{y,z} \ \|X - yz^T\|_F$$

$$\text{subject to } \|z\|_2 = 1$$

where $y \in \mathbf{R}^m$ and $z \in \mathbf{R}^n$.

Similarly, a sparsity-inducing norm such as ℓ_0 and ℓ_1 on z can be added to encourage sparse solutions. The formulation as discussed in Chapter 2 will be a non-convex

optimization problem and hence only admits an algorithm to find local optimal solutions. This approach merits discussion as the algorithm is very simple and scalable.

An important aspect of this dissertation is to study the interpretability issues of Sparse PCA solutions. We will first run Sparse PCA on data sets of various nature, illustrating how Sparse PCA brings about clearer interpretability and better visualization in real data. We will also comment on new interesting findings revealed by Sparse PCA in each case.

Then we focus on applying Sparse PCA to analysis of text data with online text news as our particular focus, showing that sparse PCA can uncover interesting topics within a large text corpora. Here a topic corresponds to a short list of terms that should be semantically coherent.

The text data we are dealing with are large-scale, typically involving around 10,000 features. However, the existing first-order algorithm for solving the semidefinite formulation, as developed in d'Aspremont et al. [9], has a computational complexity of $O(n^4\sqrt{\log n})$, with n the number of features, which is too high for such large-scale data sets. Therefore, we develop a faster block coordinate ascent algorithm with better dependence on problem size, which is another major contribution from this dissertation. The new algorithm, coupled with a rigorous feature elimination method, works pretty well in practice.

In terms of dissertation organization, we first discuss several problem formulations for Sparse PCA and describe algorithms to solve the formulations in Chapter 2. We then develop the new algorithm to solve the semidefinite formulation called DSPCA for large-scale data in Chapter 3. Finally we report numerical results on various data sets: newsgroup data, Senate voting records and stock market returns, and discuss a potentially very useful application of Sparse PCA to exploring a large text

corpora in Chapter 4. We draw conclusions and point out future research directions in Chapter 5.

Chapter 2

Sparse PCA

2.1 Introduction

While PCA is numerically easy, each factor requires computing a leading eigenvector, which can be done in $O(n^2)$ floating point operations using the Lanczos method for example (see e.g. Golub and Van Loan [15, §8.3, §9.1.1] or Saad [36] for details), Sparse PCA is a hard combinatorial problem. In fact, Moghaddam et al. [25] show that the subset selection problem for ordinary least squares, which is NP-hard (Natarajan [30]), can be reduced to a sparse generalized eigenvalue problem, of which sparse PCA is a particular instance. Sometimes factor rotation techniques are used to post-process the results from PCA and improve interpretability (see QUARTIMAX by Neuhaus and Wrigley [34], VARIMAX by Kaiser [21] or Jolliffe [17] for a discussion). Another straightforward solution is to *threshold* to zero loadings with small magnitude (Cadima and Jolliffe [6]), but outside of easy cases, the methods highlighted below always perform better in situation when only a few observations are available or when significant noise is present (Zhang et al. [42]).

A more systematic approach to the problem arose in recent years, with various

researchers proposing nonconvex algorithms (e.g., SCoTLASS by Jolliffe et al. [19], SLRA by Zhang et al. [44] or D.C. based methods Sriperumbudur et al. [39] which find modified principal components with zero loadings). The SPCA algorithm, which is based on the representation of PCA as a regression-type optimization problem (Zou et al. [45]), allows the application of the LASSO (Tibshirani [40]), a penalization technique based on the ℓ_1 norm. With the exception of simple thresholding, all the algorithms above require solving non convex problems. Recently also, d'Aspremont et al. [9] derived an ℓ_1 based semidefinite relaxation for the sparse PCA problem (1.1) with a complexity of $O(n^4\sqrt{\log n})$ for a given ρ. Moghaddam et al. [26] used greedy search and branch-and-bound methods to solve small instances of problem (1.1) exactly and get good solutions for larger ones. Each step of this greedy algorithm has complexity $O(n^3)$, leading to a total complexity of $O(n^4)$ for a full set of solutions. Moghaddam et al. [27] improve this bound in the regression/discrimination case. Journée et al. [20] use an extension of the power method to (locally) solve the problem defined here, as well as the "block" problem of finding several sparse principal components at once. Loss of orthogonality means that there is no natural method for deflating the matrix once a sparse principal component is found and Mackey [24] discusses several options, comparing the variance vs. orthogonality/sparsity tradeoffs they imply. Finally, Amini and Wainwright [2] derive explicit sample size thresholds for recovery of true sparse vector using either simple thresholding methods or semidefinite relaxations, in a spiked model for the covariance.

In this chapter, we first discuss a non-convex formulation based on low-rank approximation of matrices and describe a local but very simple and scalable algorithm for solving the problem. Then we review two semidefinite relaxations based on ℓ_1 and ℓ_0 penalizations respectively. The algorithm as developed by d'Aspremont et al. [9] for solving the semidefinite relaxation based on ℓ_1 penalization is also briefly described. This first-order algorithm has a complexity complexity of $O(n^4\sqrt{\log n})$ and

does not work well with large-scale problems, which motivates us to develop a new block coordinate ascent algorithm to be presented later in Chapter 3. Several greedy methods are also described in this chapter and we compare them with the first-order algorithm on synthetic data. We leave the discussion of the performance on real data till later in Chapter 4.

Notation. For a vector $z \in \mathbf{R}$, we let $\|z\|_1 = \sum_{i=1}^n |z_i|$ and $\|z\| = (\sum_{i=1}^n z_i^2)^{1/2}$, $\mathbf{Card}(z)$ is the cardinality of z, i.e. the number of nonzero coefficients of z, while the support I of z is the set $\{i : \ z_i \neq 0\}$ and we use I^c to denote its complement. For $\beta \in \mathbf{R}$, we write $\beta_+ = \max\{\beta, 0\}$ and for $X \in \mathbf{S}_n$ (the set of symmetric matrix of size $n \times n$) with eigenvalues λ_i, $\mathbf{Tr}(X)_+ = \sum_{i=1}^n \max\{\lambda_i, 0\}$. The vector of all ones is written $\mathbf{1}$, while the identity matrix is written $\mathbf{I}$. The diagonal matrix with the vector u on the diagonal is written $\mathbf{diag}(u)$.

2.2 A low-rank approximation approach

One way to formulate principal component analysis involves as a crucial step the approximation of a $n \times m$ data matrix M by a rank-one matrix. The problem can be formulated as the non-convex one:

$$\min_{p,q} \ \|M - pq^T\|_F^2. \tag{2.1}$$

Where $p \in \mathbf{R}^n$, and $q \in \mathbf{R}^m$. For large, sparse matrices M, the famous power iteration method is an efficient algorithm that is guaranteed to converge when the largest singular value of M is simple. The algorithm amounts to solve the above alternatively over p, q, in a "block-coordinate" fashion. The iterations are

$$p \to \frac{1}{q^T q} Mq, \ \ q \to \frac{1}{p^T p} M^T p,$$

These can be expressed in terms of normalized vectors $\tilde{p} = p/\|p\|_2$, $\tilde{q} := q/\|q\|_2$, as

$$\tilde{p} \to P(M\tilde{q}), \quad \tilde{q} \to P(M^T\tilde{p}),$$

where P is the projection on the unit circle (assigning to a non-zero vector v its scaled version $v/\|v\|_2$).

Based on the PCA formulation 2.1, one way to formulate the sparse PCA problem (see Shen and Huang [38], Mackey [24]) involves a low-rank approximation problem where the sparsity of the low-rank approximation is penalized:

$$\min_{p,q} \frac{1}{2}\|M - pq^T\|_F^2 + \lambda\|p\|_1 + \mu\|q\|_1, \tag{2.2}$$

where M is a $n \times m$ data matrix, $\|\cdot\|_F$ is the Frobenius norm, and $\mu \geq 0, \lambda \geq 0$ are parameters.

The model above results in a rank-one approximation to M (the matrix pq^T at optimum), and vectors p, q are encouraged to be sparse due to the presence of the l_1 norms, with high value of the parameters λ, μ yielding sparser results. Once sparse solutions are found, then the rows (resp. columns) in M corresponding to zero elements in p (resp. in q) are removed, and problem (2.2) is solved with the reduced matrix as input. If M is a term-by-document matrix, the above model provides sparsity in the feature space (via p) and the document space (via a "topic model" q), allowing to pinpoint a few features and a few documents that jointly "explain" data variance.

Several algorithms have been proposed for the sparse PCA problem, for example by Journée et al. [20], Shen and Huang [38], d'Aspremont et al. [10]. In practice, one algorithm that is very efficient (although it is only guaranteed to converge to a local minimum) consists in solving the above problem alternatively over p, q many times (Shen and Huang [38]). This leads to a modified power iteration method

$$\tilde{p} \to P(S_\lambda(\tilde{q})), \quad \tilde{q} \to P(S_\mu(M^T\tilde{p})),$$

where P is again the projection on the unit circle, and for $t \geq 0$, S_t is the "soft thresholding" operator (for a given vector v, $S_t(v) = \mathbf{sign}(v)\max(0, |v| - t)$, with operations acting component-wise). We can replace the soft thresholding by hard thresholding, for example zeroing out all but a fixed number of the largest elements in the vector involved.

With $\lambda = \mu = 0$ the original power iteration method for the computation of the largest singular value of M is recovered, with optimal p, q the right- and left- singular vectors of M. The presence of λ, μ modifies these singular vectors to make them sparser, while maintaining the closeness of M to its rank-one approximation. The hard-thresholding version of power iteration scales extremely well with problem size, with greatest speed increases over standard power iteration for PCA when a high degree of sparsity is asked for. This is because the vectors p, q are maintained to be extremely sparse during the iterations.

2.3 A semidefinite relaxation with ℓ_1 penalization

Starting from the variational representation of finding the first principal component $z \in \mathbf{R}^n$:

$$\max \; z^T \Sigma z$$
$$\text{subject to } \|z\|_2 = 1$$

where $\Sigma \in \mathbf{S}_n$ is the sample covariance matrix. Problem 2.3 is another way to formulate the sparse PCA problem:

$$\phi(\rho) \equiv \max_{\|z\|_2 \leq 1} z^T \Sigma z - \rho \, \mathbf{Card}(z) \tag{2.3}$$

where $\rho > 0$ is a parameter controlling sparsity. We assume without loss of generality that $\Sigma \in \mathbf{S}_n$ is positive semidefinite and that the n variables are ordered by decreasing

marginal variances, i.e. that $\Sigma_{11} \geq \ldots \geq \Sigma_{nn}$. We also assume that we are given a square root A of the matrix Σ with $\Sigma = A^T A$, where $A \in \mathbf{R}^{n\times n}$ and we denote by $a_1, \ldots, a_n \in \mathbf{R}^n$ the columns of A. Note that the problem and the algorithms for solving it are invariant by permutations of Σ and by the choice of square root A. In practice, we are very often given the data matrix A instead of the covariance Σ.

A problem that is directly related to (2.3) is that of computing a cardinality constrained maximum eigenvalue, by solving

$$\begin{array}{ll} \text{maximize} & z^T \Sigma z \\ \text{subject to} & \mathbf{Card}(z) \leq k \\ & \|z\| = 1, \end{array} \tag{2.4}$$

in the variable $z \in \mathbf{R}^n$. Of course, this problem and (2.3) are related. By weak duality, an upper bound on the optimal value of (2.4) is given by

$$\inf_{\rho \in P} \phi(\rho) + \rho k.$$

where P is the set of penalty values for which $\phi(\rho)$ has been computed. This means in particular that if a point z is provably optimal for (2.3), it is also globally optimum for (2.4) with $k = \mathbf{Card}(z)$.

Here, we briefly recall the ℓ_1 based relaxation derived by d'Aspremont et al. [9]. Following the *lifting procedure* for semidefinite relaxation described by Lovász and Schrijver [23], Alizadeh [1], Lemaréchal and Oustry [22] for example, we rewrite (2.4) as

$$\begin{array}{ll} \text{maximize} & \mathbf{Tr}(\Sigma X) \\ \text{subject to} & \mathbf{Tr}(X) = 1 \\ & \mathbf{Card}(X) \leq k^2 \\ & X \succeq 0, \ \ \mathbf{Rank}(X) = 1, \end{array} \tag{2.5}$$

in the (matrix) variable $X \in \mathbf{S}^n$. Programs (2.4) and (2.5) are equivalent, indeed if X is a solution to the above problem, then $X \succeq 0$ and $\mathbf{Rank}(X) = 1$ mean that

we have $X = xx^T$, while $\mathbf{Tr}(X) = 1$ implies that $\|x\|_2 = 1$. Finally, if $X = xx^T$ then $\mathbf{Card}(X) \leq k^2$ is equivalent to $\mathbf{Card}(x) \leq k$. We have made some progress by turning the convex maximization objective $x^T\Sigma x$ and the nonconvex constraint $\|x\|_2 = 1$ into a linear constraint and linear objective. Problem (2.5) is, however, still nonconvex and we need to relax both the rank and cardinality constraints.

Since for every $u \in \mathbf{R}^n$, $\mathbf{Card}(u) = q$ implies $\|u\|_1 \leq \sqrt{q}\|u\|_2$, we can replace the nonconvex constraint $\mathbf{Card}(X) \leq k^2$, by a weaker but convex constraint: $\mathbf{1}^T|X|\mathbf{1} \leq k$, where we exploit the property that $\|X\|_F = \sqrt{x^Tx} = 1$ when $X = xx^T$ and $\mathbf{Tr}(X) = 1$. If we drop the rank constraint, we can form a relaxation of (2.5) and (2.4) as

$$\begin{array}{ll} \text{maximize} & \mathbf{Tr}(\Sigma X) \\ \text{subject to} & \mathbf{Tr}(X) = 1 \\ & \mathbf{1}^T|X|\mathbf{1} \leq k \\ & X \succeq 0, \end{array} \tag{2.6}$$

which is a semidefinite program in the variable $X \in \mathbf{S}^n$, where k is an integer parameter controlling the sparsity of the solution. The optimal value of this program will be an upper bound on the optimal value of the variational problem in (2.4). Here, the relaxation of $\mathbf{Card}(X)$ in $\mathbf{1}^T|X|\mathbf{1}$ corresponds to a classic technique which replaces the (non-convex) cardinality or l_0 norm of a vector x with its largest convex lower bound on the unit box: $|x|$, the l_1 norm of x (see Fazel et al. [13] or Donoho and Tanner [11] for other applications).

Problem (2.6) can be interpreted as a robust formulation of the maximum eigenvalue problem, with additive, componentwise uncertainty in the input matrix Σ. We again assume Σ to be symmetric and positive semidefinite. If we consider a variation in which we penalize by the ℓ_1 norm of the matrix X instead of imposing a hard

bound, to get

$$\begin{array}{ll} \text{maximize} & \mathbf{Tr}(\Sigma X) - \rho \mathbf{1}^T |X| \mathbf{1} \\ \text{subject to} & \mathbf{Tr}(X) = 1 \\ & X \succeq 0, \end{array} \tag{2.7}$$

which is a semidefinite program in the variable $X \in \mathbf{S}^n$, where $\rho > 0$ controls the magnitude of the penalty. We can rewrite this problem as

$$\max_{X \succeq 0, \mathbf{Tr}(X)=1} \; \min_{|U_{ij}| \leq \rho} \mathbf{Tr}(X(\Sigma + U)) \tag{2.8}$$

in the variables $X \in \mathbf{S}^n$ and $U \in \mathbf{S}^n$. This yields the following dual to (2.7)

$$\begin{array}{ll} \text{minimize} & \lambda^{\max}(\Sigma + U) \\ \text{subject to} & |U_{ij}| \leq \rho, \quad i, j = 1, \ldots, n, \end{array} \tag{2.9}$$

which is a maximum eigenvalue problem with variable $U \in \mathbf{S}^n$. This gives a natural robustness interpretation to the relaxation in (2.7): it corresponds to a worst-case maximum eigenvalue computation, with componentwise bounded noise of intensity ρ imposed on the matrix coefficients.

Finally, the KKT conditions (see Boyd and Vandenberghe [4, §5.9.2]) for problem (2.7) and (2.9) are given by

$$\begin{cases} (\Sigma + U)X = \lambda^{\max}(\Sigma + U)X \\ U \circ X = \rho |X| \\ \mathbf{Tr}(X) = 1, \; X \succeq 0 \\ |U_{ij}| \leq \rho, \quad i, j = 1, \ldots, n. \end{cases} \tag{2.10}$$

Where $U \circ X$ is the Hadamard (component-wise) product of U and X. If the eigenvalue $\lambda^{\max}(\Sigma + U)$ is simple (when, for example, $\lambda^{\max}(A)$ is simple and ρ is sufficiently small), the first condition means that $\mathbf{Rank}(X) = 1$ and the semidefinite relaxation is *tight*, with in particular $\mathbf{Card}(X) = \mathbf{Card}(x)^2$ if x is the dominant eigenvector of X. When the optimal solution X is not of rank one because of degeneracy (i.e. when $\lambda^{\max}(\Sigma + U)$ has multiplicity strictly larger than one), we can truncate X as

in Alizadeh [1], Lemaréchal and Oustry [22], retaining only the dominant eigenvector x as an approximate solution to the original problem. In that degenerate scenario however, the dominant eigenvector of X is not guaranteed to be as sparse as the matrix itself.

First-order method The DSPCA code developed by d'Aspremont et al. [9] solves the dual of the penalized formulation (2.7), rewritten below as

$$\begin{array}{ll} \text{minimize} & f(U) = \lambda^{\max}(\Sigma + U) \\ \text{subject to} & |U_{ij}| \leq \rho. \end{array} \tag{2.11}$$

in the variable $U \in \mathbf{S}^n$.

The algorithm in (d'Aspremont et al. [9], Nesterov [33]) regularizes the objective $f(U)$ in (2.11), replacing it by the smooth (i.e. with Lipschitz continuous gradient) uniform approximation

$$f_\mu(U) = \mu \log \left(\mathbf{Tr} \exp((\Sigma + U)/\mu)\right) - \mu \log n.$$

Following Nesterov [31], solving the smooth problem

$$\min_{U \in \mathcal{Q}} f_\mu(U)$$

where $\mathcal{Q} = \{U \in \mathbf{S}^n, |U_{ij}| \leq \rho\}$, with $\mu = \epsilon/2\log(n)$ then produces an ϵ-approximate solution to (2.7). The key difference between the minimization scheme developed in Nesterov [31] and classical gradient minimization methods is that it is not a descent method but achieves a complexity of $O(L/N^2)$ instead of $O(1/N)$ for gradient descent, where N is the number of iterations and L the Lipschitz constant of the gradient. Furthermore, this convergence rate is provably optimal for this particular class of convex minimization problems (see Nesterov [32, Th. 2.1.13]). Thus, by sacrificing the (local) properties of descent directions, we improve the (global) complexity estimate

Algorithm 1 First-Order Algorithm.

Input: The covariance $\Sigma \in \mathbf{R}^{n\times n}$, and a parameter $\rho > 0$ controlling sparsity.

1: **for** $i = 1$ to N **do**

2: Compute $f_\mu(U_i)$ and $\nabla f_\mu(U_i)$

3: Find $Y_i = \arg\min_{Y\in\mathcal{Q}} \langle \nabla f_\mu(U_i), Y\rangle + \frac{1}{2}L\|U_i - Y\|_F^2$

4: Find $W_i = \arg\min_{W\in\mathcal{Q}} \left\{ \frac{L\|W\|_F^2}{2} + \sum_{j=0}^{N} \frac{j+1}{2}(f_\mu(U_j) + \langle \nabla f_\mu(U_j), W - U_j\rangle) \right\}$

5: Set $U_{i+1} = \frac{2}{i+3}W_i + \frac{i+1}{i+3}Y_i$

6: **end for**

Output: A matrix $U \in \mathbf{S}_n$.

by an order of magnitude. For our problem here, once the regularization parameter μ is set, the algorithm is detailed as Algorithm 1.

The algorithm has four main steps. Step one computes the (smooth) function value and gradient. The second step computes the *gradient mapping*, which matches the gradient step for unconstrained problems (see Nesterov [32, p.86]). Step three and four update an *estimate sequence* (see Nesterov [32, p.72]) of f_μ whose minimum can be computed explicitly and gives an increasingly tight upper bound on the minimum of f_μ. We now present these steps in detail for our problem (we write U for U_i and X for X_i).

Step 1. The most expensive step in the algorithm is the first, the computation of f_μ and its gradient. The function value can be reliably computed as

$$f_\mu(U) = d_{\max} + \mu \log\left(\sum_{i=1}^{n} \exp(\frac{d_i - d_{\max}}{\mu})\right) - \mu \log n.$$

where d_i are the eigenvalues of $\Sigma+U$. The gradient $\nabla f_\mu(U)$ can be computed explicitly as

$$\nabla f_\mu(U) := \exp\left((\Sigma + U)/\mu\right) / \mathbf{Tr}\left(\exp\left((\Sigma + U)/\mu\right)\right).$$

which means computing the same matrix exponential.

Step 2. This step involves a problem of the form

$$\arg\min_{Y\in\mathcal{Q}} \langle \nabla f_\mu(U), Y\rangle + \frac{1}{2}L\|U-Y\|_F^2,$$

where U is given. The above problem can be reduced to a Euclidean projection

$$\arg\min_{\|Y\|_\infty\leq 1} \|Y-V\|_F, \tag{2.12}$$

where $V = U - L^{-1}\nabla f_\mu(U)$ is given. The solution is given by

$$Y_{ij} = \mathbf{sgn}(V_{ij})\min(|V_{ij}|, 1), \quad i,j = 1,\ldots,n.$$

Step 3. The third step involves solving a Euclidean projection problem similar to (2.12), with the solution V defined by

$$V = -\frac{1}{L}\sum_{i=0}^{k}\frac{i+1}{2}\nabla f_\mu(U_i).$$

Stopping criterion We can stop the algorithm when the duality gap is smaller than ϵ:

$$\text{gap}_k = \lambda_{\max}(\Sigma + U_k) - \mathbf{Tr}\,\Sigma X_i + \mathbf{1}^T|X_i|\mathbf{1} \leq \epsilon,$$

where $X_k = \nabla f_\mu(U)$ is our current estimate of the dual variable. The above gap is necessarily non-negative, since both X_i and U_i are feasible for the primal and dual problem, respectively. This is checked periodically, for example every 100 iterations.

Complexity Overall the algorithm requires

$$O\left(\rho\frac{n\sqrt{\log n}}{\epsilon}\right) \tag{2.13}$$

iterations (d'Aspremont et al. [9], Nesterov [33]). The main step at each iteration is computing the matrix exponential $\exp((\Sigma+U)/\mu)$ (see Moler and Van Loan [28] for a comprehensive survey) at a cost of $O(n^3)$ flops.

2.4 A semidefinite relaxation with ℓ_0 penalization

We summarize here the results in d'Aspremont et al. [10]. We begin by reformulating (2.3) as a relatively simple convex maximization problem. Suppose that $\rho \geq \Sigma_{11}$. Since $z^T\Sigma z \leq \Sigma_{11}(\sum_{i=1}^n |z_i|)^2$ and $(\sum_{i=1}^n |z_i|)^2 \leq \|z\|^2 \mathbf{Card}(z)$ for all $z \in \mathbf{R}^n$, we have

$$\begin{aligned}\phi(\rho) &= \max_{\|z\|\leq 1} z^T\Sigma z - \rho\,\mathbf{Card}(z)\\ &\leq (\Sigma_{11}-\rho)\,\mathbf{Card}(z)\\ &\leq 0,\end{aligned}$$

hence the optimal solution to (2.3) when $\rho \geq \Sigma_{11}$ is $z = 0$. From now on, we assume $\rho \leq \Sigma_{11}$ in which case the inequality $\|z\| \leq 1$ is tight. We can represent the sparsity pattern of a vector z by a vector $u \in \{0,1\}^n$ and rewrite (2.3) in the equivalent form

$$\begin{aligned}\phi(\rho) &= \max_{u\in\{0,1\}^n} \lambda_{\max}(\mathbf{diag}(u)\Sigma\,\mathbf{diag}(u)) - \rho\mathbf{1}^T u\\ &= \max_{u\in\{0,1\}^n} \lambda_{\max}(\mathbf{diag}(u)A^TA\,\mathbf{diag}(u)) - \rho\mathbf{1}^T u\\ &= \max_{u\in\{0,1\}^n} \lambda_{\max}(A\,\mathbf{diag}(u)A^T) - \rho\mathbf{1}^T u,\end{aligned}$$

using the fact that $\mathbf{diag}(u)^2 = \mathbf{diag}(u)$ for all variables $u \in \{0,1\}^n$ and that for any matrix B, $\lambda_{\max}(B^TB) = \lambda_{\max}(BB^T)$. We then have

$$\begin{aligned}\phi(\rho) &= \max_{u\in\{0,1\}^n} \lambda_{\max}(A\,\mathbf{diag}(u)A^T) - \rho\mathbf{1}^T u\\ &= \max_{\|x\|=1} \ \max_{u\in\{0,1\}^n} x^TA\,\mathbf{diag}(u)A^Tx - \rho\mathbf{1}^T u\\ &= \max_{\|x\|=1} \ \max_{u\in\{0,1\}^n} \sum_{i=1}^n u_i((a_i^Tx)^2-\rho).\end{aligned}$$

Hence we finally get, after maximizing in u (and using $\max_{v\in\{0,1\}} \beta v = \beta_+$)

$$\phi(\rho) = \max_{\|x\|=1} \sum_{i=1}^n ((a_i^Tx)^2-\rho)_+, \tag{2.14}$$

which is a nonconvex problem in the variable $x \in \mathbf{R}^n$. We then select variables i such that $(a_i^Tx)^2 - \rho > 0$. Note that if $\Sigma_{ii} = a_i^Ta_i < \rho$, we must have $(a_i^Tx)^2 \leq \|a_i\|^2\|x\|^2 < \rho$ hence variable i will never be part of the optimal subset and we can remove it.

Because the variable x appears solely through $X = xx^T$, we can reformulate the problem in terms of X only, using the fact that when $\|x\| = 1$, $X = xx^T$ is equivalent to $\mathbf{Tr}(X) = 1$, $X \succeq 0$ and $\mathbf{Rank}(X) = 1$. We thus rewrite (2.14) as

$$\begin{array}{rll} \phi(\rho) = & \max. & \sum_{i=1}^n (a_i^T X a_i - \rho)_+ \\ & \text{s.t.} & \mathbf{Tr}(X) = 1, \ \mathbf{Rank}(X) = 1 \\ & & X \succeq 0. \end{array}$$

Note that because we are maximizing a convex function $\Delta_n = \{X \in \mathbf{S}_n : \ \mathbf{Tr}(X) = 1, \ X \succeq 0\}$ which is convex, the solution must be an extreme point of Δ_n (i.e. a rank one matrix), hence we can drop the rank constraint here. Unfortunately, $X \mapsto (a_i^T X a_i - \rho)_+$, the function we are *maximizing*, is convex in X and not concave, which means that the above problem is still hard. However, we show below that on rank one elements of Δ_n, it is also equal to a concave function of X, and we use this to produce a semidefinite relaxation of problem (2.3).

Proposition 2.4.1 *Let $A \in \mathbf{R}^{n\times n}$, $\rho \geq 0$ and denote by $a_1, \ldots, a_n \in \mathbf{R}^n$ the columns of A, an upper bound on*

$$\begin{array}{rll} \phi(\rho) = & \mathit{max.} & \sum_{i=1}^n (a_i^T X a_i - \rho)_+ \\ & \mathit{s.t.} & \mathbf{Tr}(X) = 1, \ X \succeq 0, \ \mathbf{Rank}(X) = 1 \end{array} \tag{2.15}$$

can be computed by solving

$$\begin{array}{rll} \psi(\rho) = & \mathit{max.} & \sum_{i=1}^n \mathbf{Tr}(X^{1/2} B_i X^{1/2})_+ \\ & \mathit{s.t.} & \mathbf{Tr}(X) = 1, \ X \succeq 0. \end{array} \tag{2.16}$$

in the variables $X \in \mathbf{S}_n$, where $B_i = a_i a_i^T - \rho \mathbf{I}$, or also

$$\begin{array}{rll} \psi(\rho) = & \mathit{max.} & \sum_{i=1}^n \mathbf{Tr}(P_i B_i) \\ & \mathit{s.t.} & \mathbf{Tr}(X) = 1, \ X \succeq 0, \ X \succeq P_i \succeq 0, \end{array} \tag{2.17}$$

which is a semidefinite program in the variables $X \in \mathbf{S}_n$, $P_i \in \mathbf{S}_n$.

Proof Let $X^{1/2}$ be the positive square root (i.e. with nonnegative eigenvalues) of a symmetric positive semi-definite matrix X. In particular, if $X = xx^T$ with $\|x\| = 1$, then $X^{1/2} = X = xx^T$, and for all $\beta \in \mathbf{R}$, βxx^T has one eigenvalue equal to β and $n-1$ equal to 0, which implies $\mathbf{Tr}(\beta xx^T)_+ = \beta_+$. We thus get

$$\begin{aligned}(a_i^T X a_i - \rho)_+ &= \mathbf{Tr}((a_i^T xx^T a_i - \rho)xx^T)_+ \\ &= \mathbf{Tr}(x(x^T a_i a_i^T x - \rho)x^T)_+ \\ &= \mathbf{Tr}(X^{1/2} a_i a_i^T X^{1/2} - \rho X)_+ = \mathbf{Tr}(X^{1/2}(a_i a_i^T - \rho \mathbf{I})X^{1/2})_+.\end{aligned}$$

For any symmetric matrix B, the function $X \mapsto \mathbf{Tr}(X^{1/2}BX^{1/2})_+$ is concave on the set of symmetric positive semidefinite matrices, because we can write it as

$$\begin{aligned}\mathbf{Tr}(X^{1/2}BX^{1/2})_+ &= \max_{\{0 \preceq P \preceq X\}} \mathbf{Tr}(PB) \\ &= \min_{\{Y \succeq B,\ Y \succeq 0\}} \mathbf{Tr}(YX),\end{aligned}$$

where this last expression is a concave function of X as a pointwise minimum of affine functions. We can now relax the original problem into a convex optimization problem by simply dropping the rank constraint, to get

$$\begin{array}{rll}\psi(\rho) \equiv & \text{max.} & \sum_{i=1}^n \mathbf{Tr}(X^{1/2} a_i a_i^T X^{1/2} - \rho X)_+ \\ & \text{s.t.} & \mathbf{Tr}(X) = 1,\ X \succeq 0,\end{array}$$

which is a convex program in $X \in \mathbf{S}_n$. Note that because B_i has at most one nonnegative eigenvalue, we can replace $\mathbf{Tr}(X^{1/2} a_i a_i^T X^{1/2} - \rho X)_+$ by $\lambda_{\max}(X^{1/2} a_i a_i^T X^{1/2} - \rho X)_+$ in the above program. Using the representation of $\mathbf{Tr}(X^{1/2}BX^{1/2})_+$ detailed above, problem (2.16) can be written as a semidefinite program

$$\begin{array}{rll}\psi(\rho) = & \text{max.} & \sum_{i=1}^n \mathbf{Tr}(P_i B_i) \\ & \text{s.t.} & \mathbf{Tr}(X) = 1,\ X \succeq 0,\ X \succeq P_i \succeq 0,\end{array}$$

in the variables $X \in \mathbf{S}_n,\ P_i \in \mathbf{S}_n$, which is the desired result.

Note that we always have $\psi(\rho) \geq \phi(\rho)$ and when the solution to the above semidefinite program has rank one, $\psi(\rho) = \phi(\rho)$ and the semidefinite relaxation (2.17) is *tight*.

This simple fact allows to derive sufficient global optimality conditions for the original sparse PCA problem. We recall in particular the following result from d'Aspremont et al. [10] which provides sufficient conditions for a particular nonzero coefficient pattern I to be globally optimal. The optimal solution x to (2.3) is then found by solving an eigenvalue problem on the principal submatrix of Σ with support I.

Proposition 2.4.2 *Let $A \in \mathbf{R}^{n\times n}$, $\rho \geq 0$, $\Sigma = A^TA$ with $a_1, \ldots, a_n \in \mathbf{R}^n$ the columns of A. Given a sparsity pattern I, setting x to be the largest eigenvector of $\sum_{i\in I} a_i a_i^T$, if there is a $\rho^* \geq 0$ such that the following conditions hold*

$$\max_{i\in I^c}(a_i^Tx)^2 < \rho^* < \min_{i\in I}(a_i^Tx)^2 \quad \textit{and} \quad \lambda_{\max}\left(\sum_{i=1}^n Y_i\right) \leq \sum_{i\in I}((a_i^Tx)^2 - \rho^*),$$

with the dual variables Y_i defined as

$$Y_i = \max\left\{0, \rho\frac{(a_i^Ta_i - \rho)}{(\rho - (a_i^Tx)^2)}\right\}\frac{(\mathbf{I}-xx^T)a_ia_i^T(\mathbf{I}-xx^T)}{\|(\mathbf{I}-xx^T)a_i\|^2}, \quad \textit{when } i \in I^c,$$

and

$$Y_i = \frac{B_ixx^TB_i}{x^TB_ix}, \quad \textit{when } i \in I,$$

then the sparsity pattern I is globally optimal for the sparse PCA problem (2.3) with $\rho = \rho^*$ and we can form an optimal solution z by solving the maximum eigenvalue problem

$$z = \operatorname*{argmax}_{\{z_{I^c}=0,\ \|z\|=1\}} z^T\Sigma z.$$

This result also provides tractable *lower bounds* on the optimal value of (2.3) whenever the solution is not optimal.

2.5 Greedy methods

We can also find good solution to problem (2.3), or improve existing solutions, using greedy methods. We first present very simple preprocessing solutions with

complexity $O(n \log n)$ and $O(n^2)$. We then recall a simple greedy algorithm with complexity $O(n^4)$. Finally, we describe an approximate greedy algorithm that computes a full set of (approximate) solutions for problem (2.3), with complexity $O(n^3)$.

Sorting and thresholding

The simplest ranking algorithm is to sort the diagonal of the matrix Σ and rank the variables by variance. This works intuitively because the diagonal is a rough proxy for the eigenvalues: the Schur-Horn theorem states that the diagonal of a matrix majorizes its eigenvalues (Horn and Johnson [16]); sorting costs $O(n \log n)$. Another quick solution is to compute the leading eigenvector of Σ and form a sparse vector by thresholding to zero the coefficients whose magnitude is smaller than a certain level. This can be done with cost $O(n^2)$.

Full greedy solution

Following Moghaddam et al. [26], starting from an initial solution of cardinality one at $\rho = \Sigma_{11}$, we can update an increasing sequence of index sets $I_k \subseteq [1, n]$, scanning all the remaining variables to find the index with maximum variance contribution.

At every step, I_k represents the set of nonzero elements (or sparsity pattern) of the current point and we can define z_k as the solution to problem (2.3) given I_k, which is:

$$z_k = \underset{\{z_{I_k^c}=0,\ \|z\|=1\}}{\operatorname{argmax}} \; z^T \Sigma z - \rho k,$$

which means that z_k is formed by padding zeros to the leading eigenvector of the submatrix Σ_{I_k, I_k}. Note that the entire algorithm can be written in terms of a factorization $\Sigma = A^T A$ of the matrix Σ, which means significant computational savings

Algorithm 2 Greedy Search Algorithm.

Input: $\Sigma \in \mathbf{R}^{n\times n}$

1: Preprocessing: sort variables by decreasing diagonal elements and permute elements of Σ accordingly.

2: Compute the Cholesky decomposition $\Sigma = A^T A$.

3: Initialization: $I_1 = \{1\}$, $x_1 = a_1/\|a_1\|$.

4: **for** $i = 1$ to k^{target} **do**

5: Compute $i_k = \text{argmax}_{i\notin I_k} \lambda_{\max}\left(\sum_{j\in I_k\cup\{i\}} a_j a_j^T\right)$.

6: Set $I_{k+1} = I_k \cup \{i_k\}$ and compute x_{k+1} as the leading eigenvector of $\sum_{j\in I_{k+1}} a_j a_j^T$.

7: **end for**

Output: Sparsity patterns I_k.

when Σ is given as a Gram matrix. The matrices Σ_{I_k,I_k} and $\sum_{i\in I_k} a_i a_i^T$ have the same eigenvalues and if z is an eigenvector of Σ_{I_k,I_k}, then $A_{I_k}z/\|A_{I_k}z\|$ is an eigenvector of $A_{I_k}A_{I_k}^T$.

Approximate greedy solution

Computing $n-k$ eigenvalues at each iteration is costly and we can use the fact that uu^T is a subgradient of $\lambda_{\max}$ at X if u is a leading eigenvector of X (Boyd and Vandenberghe [4]), to get:

$$\lambda_{\max}\left(\sum_{j\in I_k\cup\{i\}} a_j a_j^T\right) \geq \lambda_{\max}\left(\sum_{j\in I_k} a_j a_j^T\right) + (x_k^T a_i)^2, \tag{2.18}$$

which means that the variance is increasing by at least $(x_k^T a_i)^2$ when variable i is added to I_k. This provides a lower bound on the objective which does not require finding $n-k$ eigenvalues at each iteration. Then the following algorithm is obtained.

Again, at every step, I_k represents the set of nonzero elements (or sparsity pattern)

Algorithm 3 Approximate Greedy Search Algorithm.

Input: $\Sigma \in \mathbf{R}^{n \times n}$

1: Preprocessing: sort variables by decreasing diagonal elements and permute elements of Σ accordingly.

2: Compute the Cholesky decomposition $\Sigma = A^T A$.

3: Initialization: $I_1 = \{1\}$, $x_1 = a_1 / \|a_1\|$.

4: **for** $i = 1$ to k^{target} **do**

5: Compute $i_k = \text{argmax}_{i \notin I_k} (x_k^T a_i)^2$.

6: Set $I_{k+1} = I_k \cup \{i_k\}$ and compute x_{k+1} as the leading eigenvector of $\sum_{j \in I_{k+1}} a_j a_j^T$.

7: **end for**

Output: Sparsity patterns I_k.

of the current point and we can define z_k as the solution to problem (2.3) given I_k, which is:

$$z_k = \underset{\{z_{I_k^c}=0,\ \|z\|=1\}}{\text{argmax}} \quad z^T \Sigma z - \rho k,$$

which means that z_k is formed by padding zeros to the leading eigenvector of the submatrix Σ_{I_k, I_k}. Better points can be found by testing the variables corresponding to the p largest values of $(x_k^T a_i)^2$ instead of picking only the best one.

Computational complexity

The complexity of computing a greedy regularization path using the classic greedy algorithm in 2.5 is $O(n^4)$: at each step k, it computes $(n - k)$ maximum eigenvalue of matrices with size k. The approximate algorithm in 2.5 computes a full path in $O(n^3)$: the first Cholesky decomposition is $O(n^3)$, while the complexity of the k-th iteration is $O(k^2)$ for the maximum eigenvalue problem and $O(n^2)$ for computing all products $(x^T a_j)$. Also, when the matrix Σ is directly given as a Gram matrix $A^T A$

with $A \in \mathbf{R}^{q \times n}$ with $q < n$, it is advantageous to use A directly as the square root of Σ and the total complexity of getting the path up to cardinality p is then reduced to $O(p^3 + p^2 n)$ (which is $O(p^3)$ for the eigenvalue problems and $O(p^2 n)$ for computing the vector products).

2.6 Numerical results

In this section, we compare the performance of various algorithms on synthetic data. We will discuss the performance on real data later in Chapter 4 .

2.6.1 Statistical consistency vs. computational complexity

As we hinted above, very simple methods such as thresholding or greedy algorithms often perform well enough on simple data sets, while obtaining good statistical fidelity on more complex (or random) data sets requires more complex algorithms. This is perfectly illustrated by the results in Amini and Wainwright [2] on a spiked covariance model. To summarize these results, suppose that the sample covariance matrix $\hat{\Sigma} \in \mathbf{S}_n$ is a noisy estimate of the true population covariance $\Sigma \in \mathbf{S}_n$ with $\hat{\Sigma} = \Sigma + \Delta$ where Δ is a noise matrix, suppose also that the leading eigenvector of the true covariance is sparse with cardinality k. Under some assumptions on the noise component Δ, Amini and Wainwright [2] show that when the ambient dimension n, the number of observations m and the number k of nonzero components in the leading eigenvector all scale to infinity, and when the ratio

$$\theta_{\text{thres}} = \frac{m}{k^2 \log(n-k)}$$

is above some critical value, then simply thresholding the diagonal of the sample covariance matrix will recover the exact support of the leading eigenvector of Σ with

probability tending to one. On the other hand, simple thresholding fails with probability one when this ratio is below a certain value. Furthermore, Amini and Wainwright [2] show that when

$$\theta_{\text{sdp}} = \frac{m}{k \log(n-k)}$$

is above some critical value, the solution of the semidefinite relaxation in Section 2.3 (if tight) will recover the exact support of the leading eigenvector of Σ with probability tending to one. On the other hand, the semidefinite relaxation fails with probability one when this ratio is below a certain value. They also show that the semidefinite programing relaxation in Section 2.3 is statistically optimal, meaning that no other method (even combinatorial ones) can recover the true support using fewer samples (up to a constant factor). This result clearly illustrates a tradeoff between statistical fidelity on one side and computational complexity on the other. In the spiked model, the semidefinite relaxation requires $O(1/k)$ fewer samples than simply thresholding the diagonal to recover the true support of the leading eigenvector of Σ, but its complexity is much higher than that of the thresholding strategy.

We can further illustrate this behavior on a simple numerical example. Suppose we are given a sample covariance $\hat{\Sigma} \in \mathbf{S}_n$ coming from a "spiked" model of covariance similar to that in Amini and Wainwright [2], with

$$\hat{\Sigma} = uu^T + VV^T/\sqrt{m}$$

where $u \in \mathbf{R}^n$ is the true sparse leading eigenvector, with $\mathbf{Card}(u) = k$, $V \in \mathbf{R}^{n \times m}$ is a noise matrix with $V_{ij} \sim \mathcal{N}(0,1)$ and m is the number of observations. We compare the performance of the simple thresholding method (on the leading eigenvector of regular PCA here) with that of the semidefinite relaxation when recovering the support of u for various values of the number of samples. Our point here is that, while variance versus cardinality is a direct way of comparing the performance of sparse PCA algorithms, accurate recovery of the support is often a far more impor-

tant objective. Many methods produce similar variance levels given a limited budget of nonzero components, but their performance in recovering the true support is often markedly different.

In Figure 2.1 on the left we compare ROC curves when recovering the support of u in the spiked model above using thresholded PCA, the approximate and full greedy algorithms in d'Aspremont et al. [10] and semidefinite relaxation (DSPCA). On the right, we plot Area Under ROC as the number of samples increase. As expected, we observe that the semidefinite relaxation performs much better when only a limited number of observations are available (m small).

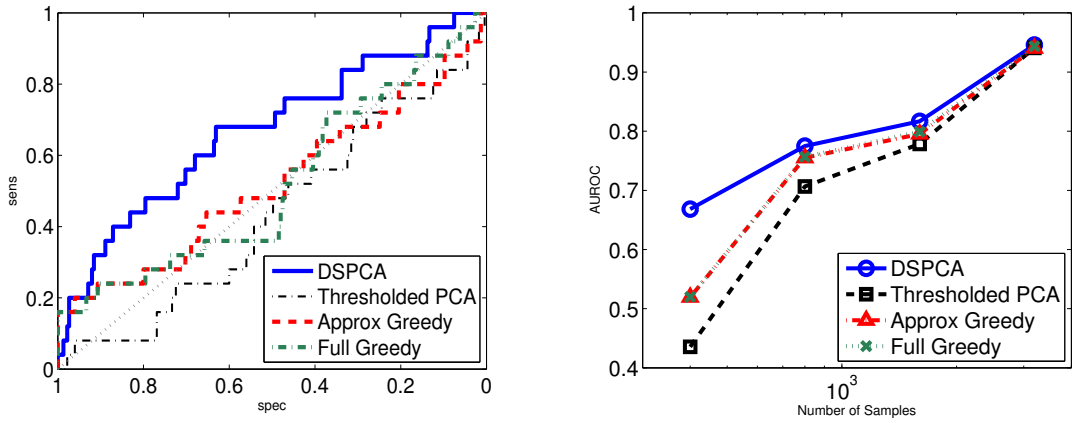

Figure 2.1: *Left:* ROC curves when recovering the support of u in the spiked model using thresholding, approximate and exact greedy algorithms and the semidefinite relaxation (DSPCA) in Section 2.3 in the spiked model when $n = 250$, $m = 400$ and $k = 25$. *Right:* Area Under ROC (AUROC) versus number of samples m.

2.6.2 Random matrices

Sparse eigenvalues of random matrices play a central role in characterizing the performance of ℓ_1 decoders in compressed sensing applications. Testing the Restricted

Isometry Property (RIP) in Candès and Tao [7] amounts to bounding the maximum and minimum eigenvalues of a Gram matrix. Here, we compute the upper and lower bounds on sparse eigenvalues produced using various algorithms. We pick the data matrix to be small enough so that computing sparse eigenvalues by exhaustive search is numerically feasible. In Figure 2.2, we plot the maximum sparse eigenvalue versus cardinality, obtained using exhaustive search (solid line), the approximate greedy (dotted line) and fully greedy (dashed line) algorithms. We also plot the upper bounds obtained by minimizing the gap of a rank one solution (squares), by solving the semidefinite relaxation in §2.4 explicitly (stars) and by solving the DSPCA dual (diamonds). On the left, we use a matrix $\Sigma = F^T F$ with F Gaussian. On the right, $\Sigma = uu^T/\|u\|^2 + 2V^T V$, where $u_i = 1/i,\ i = 1, \ldots, n$ and V is matrix with coefficients uniformly distributed in $[0, 1]$. Almost all algorithms are provably optimal in the noisy rank one case (as well as in many example arising from "natural data"), while Gaussian random matrices are harder. Note however, that the duality gap between the semidefinite relaxations and the optimal solution is very small in both cases, while our bounds based on greedy solutions are not as good. Overall, while all algorithms seem to behave similarly on "natural" or easy data sets, only numerically expensive relaxations produce good bounds on the random matrices used in compressed sensing applications.

2.7 Summary

In this chapter, we have reviewed several formulations for the *single factor* sparse PCA problem, as well as the first-order algorithms and a few greedy methods. We also compare performance of the various methods on synthetical data. When data is noisy, the semidefinite relaxation called DSPCA usually reports the best perfor-

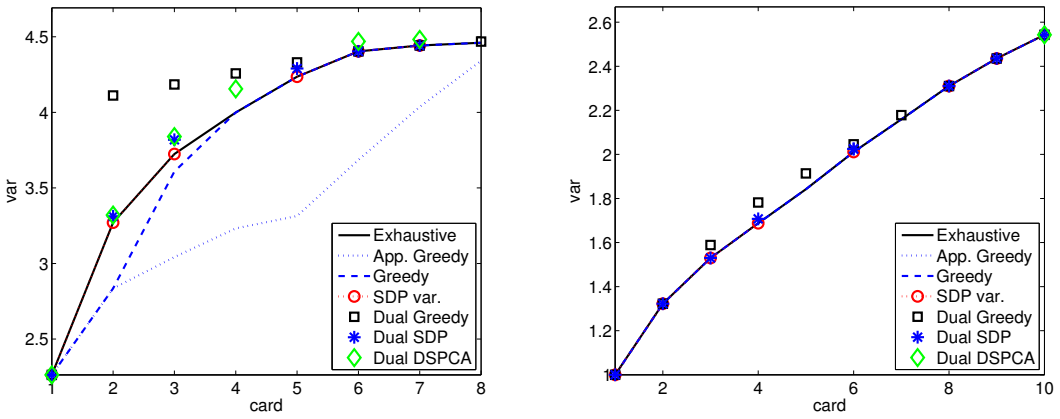

Figure 2.2: Upper and lower bound on sparse maximum eigenvalues. We plot the maximum sparse eigenvalue versus cardinality, obtained using exhaustive search (solid line), the approximate greedy (dotted line) and fully greedy (dashed line) algorithms. We also plot the upper bounds obtained by minimizing the gap of a rank one solution (squares), by solving the ℓ_0 semidefinite relaxation explicitly (stars) and by solving the DSPCA dual ℓ_1 relaxation (diamonds). *Left:* On a matrix $F^T F$ with F Gaussian. *Right:* On a sparse rank one plus noise matrix.

mance. Unfortunately, the first-order algorithm for solving DSPCA has the worst computational complexity and converges slow in practice. This motivates us to develop a faster algorithm for solving DSPCA, with better dependence on problem size in Chapter 3.

Chapter 3

Large-Scale Sparse PCA

3.1 Introduction

Sparse PCA not only brings better interpretation (d'Aspremont et al. [9]), but also provides statistical regularization (Amini and Wainwright [2]) when the number of samples is less than the number of features. Various researchers have proposed different formulations and algorithms for this problem, ranging from ad-hoc methods such as factor rotation techniques by Jolliffe [17] and simple thresholding by Cadima and Jolliffe [6], to greedy algorithms by Moghaddam et al. [26] and d'Aspremont et al. [10]. Other algorithms include SCoTLASS by Jolliffe et al. [19], SPCA by Zou et al. [45], the regularized SVD method by Shen and Huang [38] and the generalized power method by Journée et al. [20]. These algorithms are based on non-convex formulations, and may only converge to a local optimum. The ℓ_1-norm based semidefinite relaxation DSPCA, as introduced by d'Aspremont et al. [9] , does guarantee global convergence and as such, is an attractive alternative to local methods. In fact, it has been shown by d'Aspremont et al. [8], Amini and Wainwright [2], Zhang et al. [42] that simple ad-hoc methods, the greedy, SCoTLASS and SPCA algorithms, of-

ten underperform DSPCA. However, the first-order algorithm for solving DSPCA, as developed in d'Aspremont et al. [9], has a computational complexity of $O(n^4\sqrt{\log n})$, with n the number of features, which is too high for many large-scale data sets. At first glance, this complexity estimate indicates that solving sparse PCA is much more expensive than PCA, since we can compute one principal component with a complexity of $O(n^2)$.

In this chapter we show that solving DSPCA is in fact computationally easier than PCA, and hence can be applied to very large-scale data sets. To achieve that, we first view DSPCA as an approximation to a harder, cardinality-constrained optimization problem. Based on that formulation, we describe a safe feature elimination method for that problem, which leads to an often important reduction in problem size, prior to solving the problem. Then we develop a block coordinate ascent algorithm, with a computational complexity of $O(n^3)$ to solve DSPCA, which is much faster than the first-order algorithm proposed by d'Aspremont et al. [9] . Finally, we observe that real data sets typically allow for a dramatic reduction in problem size as afforded by our safe feature elimination result. Now the comparison between sparse PCA and PCA becomes $O(\hat{n}^3)$ v.s. $O(n^2)$ with $\hat{n} \ll n$, which can make sparse PCA surprisingly easier than PCA (Zhang and El Ghaoui [43]).

In Section 3.2, we relate the ℓ_1-norm based DSPCA formulation to an approximation to the ℓ_0-norm based formulation and highlight the safe feature elimination mechanism as a powerful pre-processing technique. We use Section 3.3, to present our fast block coordinate ascent algorithm. Finally, in Section 3.4,, we demonstrate the efficiency of our approach on two large data sets, each one containing more than 100,000 features.

Notation. $\mathcal{R}(Y)$ denotes the range of matrix Y, and $Y^\dagger$ its pseudo-inverse. The notation log refers to the extended-value function, with $\log x = -\infty$ if $x \leq 0$.

3.2 Safe Feature Elimination

Primal problem. Given a $n \times n$ positive-semidefinite matrix Σ, the ℓ_1-norm based DSPCA formulation introduced in d'Aspremont et al. [9] and reviewed in Chapter 2 is :

$$\phi = \max_{Z} \ \mathbf{Tr}\, \Sigma Z - \lambda \|Z\|_1 \ : \ Z \succeq 0, \ \ \mathbf{Tr}\, Z = 1 \tag{3.1}$$

where $\lambda \geq 0$ is a parameter encouraging sparsity. Without loss of generality we may assume that $\Sigma \succ 0$. This is true because we can add an arbitrary multiple of identity matrix to Σ, without changing the optimal solution to the above optimization problem.

An upper bound on λ. Without loss of generality, we can impose an upper bound on λ, as follows.

Assume first $\lambda \geq \sigma_{\max} := \max_{1 \leq i \leq n} \Sigma_{ii}$. Then the problem has a simple solution. Indeed, in that case, $Z = e_j e_j^T$ is optimal, with e_j the j-th unit vector, and j is any index such that $\Sigma_{jj} = \sigma_{\max}$.

This can be seen from the dual problem

$$\phi = \min_{U} \ \lambda_{\max}(\Sigma + U) \ : \ U = U^T, \ \ \|U\|_\infty \leq \lambda.$$

Since $\Sigma \succeq 0$, we have $|\Sigma_{kl}| \leq \sqrt{\Sigma_{kk}\Sigma_{ll}} \leq \sigma_{\max}$ for every k, l. Now define the matrix $U = U^T \in \mathbf{R}^{n \times n}$ with elements

$$U_{kl} := \begin{cases} -\Sigma_{kl} & (k,l) \neq (j,j) \\ -\lambda & k = l = j \end{cases}$$

By construction, U is feasible for the dual problem, so that

$$\phi \leq \lambda_{\max}(\Sigma + U) = \Sigma_{jj} - \lambda = \sigma_{\max} - \lambda.$$

This upper bound is attained with the primal feasible point $Z = e_j e_j^T$, which shows that Z is optimal for the primal problem (3.1).

In the sequel we can thus assume $\lambda < \sigma_{\max}$. In turn, this guarantees that $\phi > 0$, which is obtained by plugging in $Z = e_j e_j^T$ in the objective of the primal problem (3.1).

Problem (3.1) is in fact a relaxation to a PCA problem with a penalty on the cardinality of the variable:

$$\psi = \max_x \; x^T \Sigma x - \lambda \|x\|_0 \;:\; \|x\|_2 = 1 \tag{3.2}$$

Where $\|x\|_0$ denotes the cardinality (number of non-zero elemements) in x. This can be seen by first writing problem (3.2) as:

$$\max_Z \; \mathbf{Tr}\, \Sigma Z - \lambda \sqrt{\|Z\|_0} \;:\; Z \succeq 0, \;\; \mathbf{Tr}\, Z = 1, \mathbf{Rank}(Z) = 1$$

where $\|Z\|_0$ is the cardinality (number of non-zero elements) of Z.

By observing $\|Z\|_1 \leq \sqrt{\|Z\|_0}\, \|Z\|_F = \sqrt{\|Z\|_0}$ here, we obtain the relaxation

$$\max_Z \; \mathbf{Tr}\, \Sigma Z - \lambda \|Z\|_1 \;:\; Z \succeq 0, \mathbf{Tr}\, Z = 1, \mathbf{Rank}(Z) = 1$$

Further drop the rank constraint, leading to problem (3.1).

By viewing problem (3.1) as a convex approximation to the non-convex problem (3.2), we can leverage the safe feature elimination theorem first presented in d'Aspremont et al. [10] and El Ghaoui [12] for problem (3.2):

Theorem 3.2.1 *Let $\Sigma = A^T A$, where $A = (a_1, \ldots, a_n) \in \mathbf{R}^{m \times n}$. We have*

$$\psi = \max_{\|\xi\|_2 = 1} \sum_{i=1}^n ((a_i^T \xi)^2 - \lambda)_+.$$

An optimal non-zero pattern corresponds to indices i with $\lambda < (a_i^T \xi)^2$ at optimum.

We observe that the i-th feature is absent at optimum if $(a_i^T \xi)^2 \leq \lambda$ for every ξ, $\|\xi\|_2 = 1$. Hence, we can *safely* remove feature $i \in \{1, \ldots, n\}$ if

$$\Sigma_{ii} = a_i^T a_i < \lambda \tag{3.3}$$

A few remarks are in order. First, if we are interested in solving problem (3.1) as a relaxation to problem (3.2), we first calculate and rank all the feature variances, which takes $O(nm)$ and $O(n\log(n))$ respectively. Then we can safely eliminate any feature with variance less than λ. Second, the elimination criterion above is conservative. However, when looking for extremely sparse solutions, applying this safe feature elimination test with a large λ can dramatically reduce problem size and lead to huge computational savings, as will be demonstrated empirically in Section 3.4. Third, in practice, when PCA is performed on large data sets, some similar variance-based criteria is routinely employed to bring problem sizes down to a manageable level. This purely heuristic practice has a rigorous interpretation in the context of Sparse PCA, as the above theorem states explicitly the features that can be safely discarded.

3.3 Block Coordinate Ascent Algorithm

The first-order algorithm developed in d'Aspremont et al. [9] to solve problem (3.1) has a computational complexity of $O(n^4\sqrt{\log n})$. With a theoretical convergence rate of $O(\frac{1}{\epsilon})$, the DSPCA algorithm does not converge fast in practice. In this section, we develop a block coordinate ascent algorithm with better dependence on problem size ($O(n^3)$), which in practice converges much faster.

Failure of a direct method. We seek to apply a "row-by-row" algorithm by which we update each row/column pair, one at a time. This algorithm appeared in the specific context of sparse covariance estimation in O.Banerjee et al. [35], and extended to a large class of SDPs in Wen et al. [41]. Precisely, it applies to problems of the form

$$\min_X \; f(X) - \beta \log\det X \;\; : \;\; L \le X \le U, \;\; X \succ 0, \tag{3.4}$$

where $X = X^T$ is a $n \times n$ matrix variable, L, U impose component-wise bounds on X, f is convex, and $\beta > 0$.

However, if we try to update the row/columns of Z in problem (3.1), the trace constraint will imply that we never modify the diagonal elements of Z. Indeed at each step, we update only one diagonal element, and it is entirely fixed given all the other diagonal elements. The row-by-row algorithm does not directly work in that case, nor in general for SDPs with equality constraints. Wen et al. [41] propose an augmented Lagrangian method to deal with such constraints, with a complication due to the choice of appropriate penalty parameters. In our case, we can apply a technique resembling the augmented Lagrangian technique, without this added complication. This is due to the homogeneous nature of the objective function and of the conic constraint. Thanks to the feature elimination result (Thm. 3.2.1), we can always assume without loss of generality that $\lambda < \sigma_{\min}^2 := \min_{1 \leq i \leq n} \Sigma_{ii}$.

Direct augmented Lagrangian technique. We can express problem (3.1) as

$$\frac{1}{2}\phi^2 = \max_X \ \mathbf{Tr}\, \Sigma X - \lambda \|X\|_1 - \frac{1}{2}(\mathbf{Tr}\, X)^2 \ : \ X \succeq 0. \tag{3.5}$$

This expression results from the change of variable $X = \gamma Z$, with $\mathbf{Tr}\, Z = 1$, and $\gamma \geq 0$. Optimizing over $\gamma \geq 0$, and exploiting $\phi > 0$ (which comes from our assumption that $\lambda < \sigma_{\min}^2$), leads to the result, with the optimal scaling factor γ equal to ϕ. An optimal solution Z^* to (3.1) can be obtained from an optimal solution X^* to the above, via $Z^* = X^*/\phi$. (In fact, we have $Z^* = X^*/\,\mathbf{Tr}(X^*)$.)

Proof Let $X = \gamma Z$, with $\mathbf{Tr}\, Z = 1$, and $\gamma \geq 0$

$$\begin{aligned}
\text{Problem (3.5)} \quad &\Leftrightarrow \quad \max_{Z,\gamma} \ \mathbf{Tr}\, \Sigma \gamma Z - \rho \|\gamma Z\|_1 - \frac{1}{2}(\mathbf{Tr}\, \gamma Z)^2 \ : \ Z \succeq 0, \mathbf{Tr}\, Z = 1, \gamma \geq 0 \\
&\Leftrightarrow \quad \max_{Z,\gamma} \ \gamma\, \mathbf{Tr}\, \Sigma Z - \gamma \rho \|Z\|_1 - \frac{1}{2}\gamma^2 (\mathbf{Tr}\, Z)^2 \ : \ Z \succeq 0, \mathbf{Tr}\, Z = 1, \gamma \geq 0 \\
&\Leftrightarrow \quad \max_{Z,\gamma} \ \gamma(\mathbf{Tr}\, \Sigma Z - \rho \|Z\|_1) - \frac{1}{2}\gamma^2 \ : \ Z \succeq 0, \mathbf{Tr}\, Z = 1, \gamma \geq 0
\end{aligned}$$

Optimize over $\gamma \geq 0$ only, and then the above objective is just a concave quadratic function of γ. We can obtain the optimal scaling γ^* by setting the derivative to zero if we ignore the constraint $\gamma \geq 0$

$$\gamma^* = \phi = \max_Z \ \mathbf{Tr}\, \Sigma Z - \rho \|Z\|_1 \ : \ Z \succeq 0, \mathbf{Tr}\, Z = 1$$

In fact, ϕ is still the optimal scaling factor even if we take the constraint $\gamma \geq 0$ into account, because we know $\phi > 0$ and hence $\gamma^* > 0$ satisfying the constraint.

Then the optimal value is $\phi^2 - \frac{1}{2}\phi^2 = \frac{1}{2}\phi^2$. That is

$$\frac{1}{2}\phi^2 = \max_X \ \mathbf{Tr}\, \Sigma X - \lambda \|X\|_1 - \frac{1}{2}(\mathbf{Tr}\, X)^2 \ : \ X \succeq 0.$$

To apply the row-by-row method to problem (3.5), we need to consider a variant of it, with a strictly convex objective. That is, we address the problem

$$\max_X \ \mathbf{Tr}\, \Sigma X - \lambda \|X\|_1 - \frac{1}{2}(\mathbf{Tr}\, X)^2 + \beta \log\det X, \ : \ X \succ 0, \tag{3.6}$$

where $\beta > 0$ is a penalty parameter. SDP theory ensures that if $\beta = \epsilon/n$, then a solution to the above problem is ϵ-suboptimal for the original problem (Boyd and Vandenberghe [5]).

Optimizing over one row/column. Without loss of generality, we consider the problem of updating the last row/column of the matrix variable X. Partition the latter and the covariance matrix S as

$$X = \begin{pmatrix} Y & y \\ y^T & x \end{pmatrix}, \ \Sigma = \begin{pmatrix} S & s \\ s^T & \sigma \end{pmatrix},$$

where $Y, S \in \mathbf{R}^{(n-1)\times(n-1)}$, $y, s \in \mathbf{R}^{n-1}$, and $x, \sigma \in \mathbf{R}$. We are considering the problem above, where Y is fixed, and $(y, x) \in \mathbf{R}^n$ is the variable. We use the notation $t := \mathbf{Tr}\, Y$.

The conic constraint $X \succ 0$ translates as $y^T Y^\dagger y \leq x$, $y \in \mathcal{R}(Y)$, where $\mathcal{R}(Y)$ is the range of the matrix Y. We obtain the sub-problem

$$\varphi := \max_{x,y} \begin{pmatrix} 2(y^T s - \lambda \|y\|_1) + (\sigma - \lambda)x - \frac{1}{2}(t+x)^2 \\ +\beta \log(x - y^T Y^\dagger y) \end{pmatrix} \; : \; y \in \mathcal{R}(Y). \tag{3.7}$$

Simplifying the sub-problem. We can simplify the above problem, in particular, avoid the step of forming the pseudo-inverse of Y, by taking the dual of problem (3.7).

Using the conjugate relation, valid for every $\eta > 0$:

$$\log \eta + 1 = \min_{z>0} \; z\eta - \log z,$$

and with $f(x) := (\sigma - \lambda)x - \frac{1}{2}(t+x)^2$, we obtain

$$\begin{aligned} \varphi + \beta &= \max_{y \in \mathcal{R}(Y)} 2(y^T s - \lambda \|y\|_1) + f(x) + \beta \min_{z>0} \left(z(x - y^T Y^\dagger y) - \log z \right) \\ &= \min_{z>0} \max_{y \in \mathcal{R}(Y)} 2(y^T s - \lambda \|y\|_1 - \beta z y^T Y^\dagger y) + \max_x \left(f(x) + \beta z x \right) - \beta \log z \\ &= \min_{z>0} h(z) + 2g(z) \end{aligned}$$

where, for $z > 0$, we define

$$\begin{aligned} h(z) &:= -\beta \log z + \max_x \left(f(x) + \beta z x \right) \\ &= -\beta \log z + \max_x \left((\sigma - \lambda + \beta z)x - \frac{1}{2}(t+x)^2 \right) \\ &= -\frac{1}{2}t^2 - \beta \log z + \max_x \left((\sigma - \lambda - t + \beta z)x - \frac{1}{2}x^2 \right) \\ &= -\frac{1}{2}t^2 - \beta \log z + \frac{1}{2}(\sigma - \lambda - t + \beta z)^2 \end{aligned}$$

with the following relationship at optimum:

$$x = \sigma - \lambda - t + \beta z. \tag{3.8}$$

In addition,

$$\begin{aligned} g(z) &:= \max_{y \in \mathcal{R}(Y)} \; y^T s - \lambda \|y\|_1 - \frac{\beta z}{2}(y^T Y^\dagger y) \\ &= \max_{y \in \mathcal{R}(Y)} \; y^T s + \min_{v \,:\, \|v\|_\infty \le \lambda} y^T v - \frac{\beta z}{2}(y^T Y^\dagger y) \\ &= \min_{v \,:\, \|v\|_\infty \le \lambda} \; \max_{y \in \mathcal{R}(Y)} \; (y^T(s+v) - \frac{\beta z}{2}(y^T Y^\dagger y)) \\ &= \min_{u \,:\, \|u-s\|_\infty \le \lambda} \; \max_{y \in \mathcal{R}(Y)} \; (y^T u - \frac{\beta z}{2}(y^T Y^\dagger y)) \\ &= \min_{u \,:\, \|u-s\|_\infty \le \lambda} \; \frac{1}{2\beta z} u^T Y u. \end{aligned}$$

with the following relationship at optimum:

$$y = \frac{1}{\beta z} Y u. \tag{3.9}$$

Putting all this together, we obtain the dual of problem (3.7): with $\varphi' := \varphi + \beta + \frac{1}{2}t^2$, and $c := \sigma - \lambda - t$, we have

$$\varphi' = \min_{u,z} \; \frac{1}{\beta z} u^T Y u - \beta \log z + \frac{1}{2}(c + \beta z)^2 \;\; : \;\; z > 0, \;\; \|u - s\|_\infty \le \lambda.$$

Since β is small, we can avoid large numbers in the above, with the change of variable $\tau = \beta z$:

$$\varphi' - \beta \log \beta = \min_{u,\tau} \; \frac{1}{\tau} u^T Y u - \beta \log \tau + \frac{1}{2}(c + \tau)^2 \;\; : \;\; \tau > 0, \;\; \|u - s\|_\infty \le \lambda. \tag{3.10}$$

Solving the sub-problem. Problem (3.10) can be further decomposed into two stages.

First, we solve the box-constrained QP

$$R^2 := \min_u \; u^T Y u \;\; : \;\; \|u - s\|_\infty \le \lambda, \tag{3.11}$$

using a simple coordinate descent algorithm to exploit sparsity of Y. Without loss of generality, we consider the problem of updating the first coordinate of u. Partition

u, Y and s as

$$u = \begin{pmatrix} \eta \\ \hat{u} \end{pmatrix}, \quad Y = \begin{pmatrix} y_1 & \hat{y}^T \\ \hat{y} & \hat{Y} \end{pmatrix}, \quad s = \begin{pmatrix} s_1 \\ \hat{s} \end{pmatrix},$$

Where, $\hat{Y} \in \mathbf{R}^{(n-2)\times(n-2)}$, $\hat{u}, \hat{y}, \hat{s} \in \mathbf{R}^{n-2}$, $y_1, s_1 \in \mathbf{R}$ are all fixed, while $\eta \in \mathbf{R}$ is the variable. We obtain the subproblem

$$\min_{\eta} \; y_1\eta^2 + (2\hat{y}^T\hat{u})\eta \;:\; \|\eta - s_1\| \leq \lambda \tag{3.12}$$

for which we can solve for η analytically using the formula given below.

$$\eta = \begin{cases} -\frac{\hat{y}^T\hat{u}}{y_1} & \text{if } \|s_1 + \frac{\hat{y}^T\hat{u}}{y_1}\| \leq \lambda,\ y_1 > 0, \\ s_1 - \lambda & \text{if } -\frac{\hat{y}^T\hat{u}}{y_1} < s_1 - \lambda,\ y_1 > 0 \text{ or if } \hat{y}^T\hat{u} > 0,\ y_1 = 0, \\ s_1 + \lambda & \text{if } -\frac{\hat{y}^T\hat{u}}{y_1} > s_1 + \lambda,\ y_1 > 0 \text{ or if } \hat{y}^T\hat{u} <= 0,\ y_1 = 0. \end{cases} \tag{3.13}$$

Next, we set τ by solving the one-dimensional problem:

$$\min_{\tau>0} \; \frac{R^2}{\tau} - \beta\log\tau + \frac{1}{2}(c+\tau)^2.$$

The above can be reduced to a bisection problem over τ, or by solving a polynomial equation of degree 3.

Obtaining the primal variables. Once the above problem is solved, we can obtain the primal variables y, x, as follows. Using formula (3.9), with $\beta z = \tau$, we set $y = \frac{1}{\tau}Yu$. For the diagonal element x, we use formula (3.8): $x = c + \tau = \sigma - \lambda - t + \tau$.

Algorithm summary. We summarize the above derivations in Algorithm 4. Notation: for any symmetric matrix $A \in \mathbf{R}^{n\times n}$, let $A_{\setminus i \setminus j}$ denote the matrix produced by removing row i and column j. Let A_j denote column j (or row j) with the diagonal element A_{jj} removed.

Algorithm 4 Block Coordinate Ascent Algorithm

Input: The covariance matrix Σ, and a parameter $\rho > 0$.

1: Set $X^{(0)} = I$

2: **repeat**

3: **for** $j = 1$ to n **do**

4: Let $X^{(j-1)}$ denote the current iterate. Solve the box-constrained quadratic program

$$R^2 := \min_u \; u^T X^{(j-1)}_{\backslash j \backslash j} u \;\; : \;\; \|u - \Sigma_j\|_\infty \leq \lambda$$

using the coordinate descent algorithm

5: Solve the one-dimensional problem

$$\min_{\tau > 0} \; \frac{R^2}{\tau} - \beta \log \tau + \frac{1}{2}(\Sigma_{jj} - \lambda - \mathbf{Tr}\, X^{(j-1)}_{\backslash j \backslash j} + \tau)^2$$

using a bisection method, or by solving a polynomial equation of degree 3.

6: First set $X^{(j)}_{\backslash j \backslash j} = X^{(j-1)}_{\backslash j \backslash j}$, and then set both $X^{(j)}$'s column j and row j using

$$X^{(j)}_j = \frac{1}{\tau} X^{(j-1)}_{\backslash j \backslash j} u$$

$$X^{(j)}_{jj} = \Sigma_{jj} - \lambda - \mathbf{Tr}\, X^{(j-1)}_{\backslash j \backslash j} + \tau$$

7: **end for**

8: Set $X^{(0)} = X^{(n)}$

9: **until** convergence

Convergence and complexity. Our algorithm solves DSPCA by first casting it to problem (3.6), which is in the general form (3.4). Therefore, the convergence result from Wen et al. [41] readily applies and hence every limit point that our block coordinate ascent algorithm converges to is the global optimizer. The simple coordinate descent algorithm solving problem (3.11) only involves a vector product and can take sparsity in Y easily. To update each column/row takes $O(n^2)$ and there are n such

columns/rows in total. Therefore, our algorithm has a computational complexity of $O(Kn^3)$, where K is the number of sweeps through columns. In practice, K is fixed at a number independent of problem size (typically $K = 5$). Hence our algorithm has better dependence on the problem size compared to $O(n^4\sqrt{\log n})$ required of the first order algorithm developed in d'Aspremont et al. [9] .

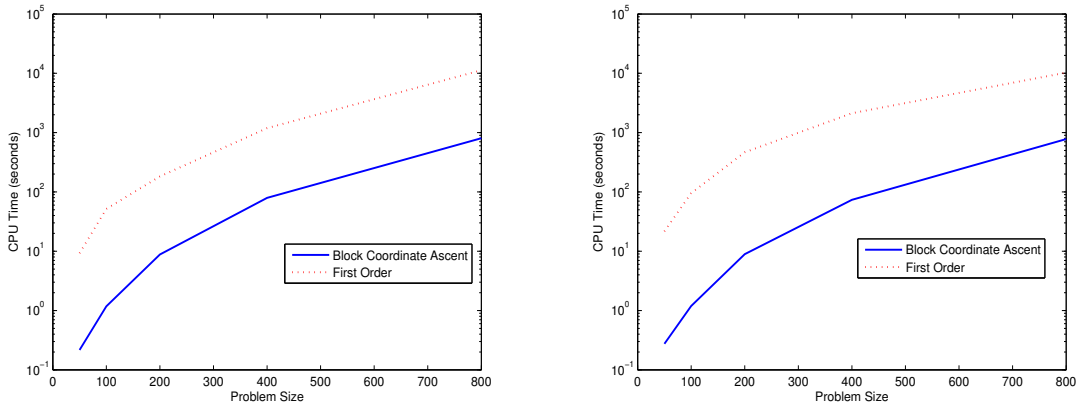

Figure 3.1: Speed comparisons between Block Coordinate Ascent and First-Order

Fig 3.1 shows that our algorithm converges much faster than the first order algorithm. On the left, both algorithms are run on a covariance matrix $\Sigma = F^T F$ with F Gaussian. On the right, the covariance matrix comes from a "spiked model" similar to that in Amini and Wainwright [2], with $\Sigma = uu^T + VV^T/m$, where $u \in \mathbf{R}^n$ is the true sparse leading eigenvector, with $\mathbf{Card}(u) = 0.1n$, $V \in \mathbf{R}^{n\times m}$ is a noise matrix with $V_{ij} \sim \mathcal{N}(0, 1)$ and m is the number of observations.

3.4 Numerical Examples

In this section, we analyze two publicly available large data sets, the NYTimes news articles data and the PubMed abstracts data, available from the UCI Machine

Learning Repository (Frank and Asuncion [14]). Both text collections record word occurrences in the form of bag-of-words. The NYTtimes text collection contains $300,000$ articles and has a dictionary of $102,660$ unique words, resulting in a file of size 1 GB. The even larger PubMed data set has $8,200,000$ abstracts with $141,043$ unique words in them, giving a file of size 7.8 GB. These data matrices are so large that we cannot even load them into memory all at once, which makes even the use of classical PCA difficult. However with the pre-processing technique presented in Section 3.2 and the block coordinate ascent algorithm developed in Section 3.3, we are able to perform sparse PCA analysis of these data, also thanks to the fact that variances of words decrease drastically when we rank them as shown in Fig 3.2. Note that the feature elimination result only requires the computation of each feature's variance, and that this task is easy to parallelize.

By doing sparse PCA analysis of these text data, we hope to find interpretable principal components that can be used to summarize and explore the large corpora. Therefore, we set the target cardinality for each principal component to be 5. As we run our algorithm with a coarse range of λ to search for a solution with the given cardinality, we might end up accepting a solution with cardinality close, but not necessarily equal to, 5, and stop there to save computational time.

The top 5 sparse principal components are shown in Table 3.1 for NYTimes and in Table 3.2 for PubMed. Clearly the first principal component for NYTimes is about business, the second one about sports, the third about U.S., the fourth about politics and the fifth about education. Bear in mind that the NYTimes data from UCI Machine Learning Repository "have no class labels, and for copyright reasons no filenames or other document-level metadata" (Frank and Asuncion [14]). The sparse principal components still unambiguously identify and perfectly correspond to the topics used by *The New York Times* itself to classify articles on its own website.

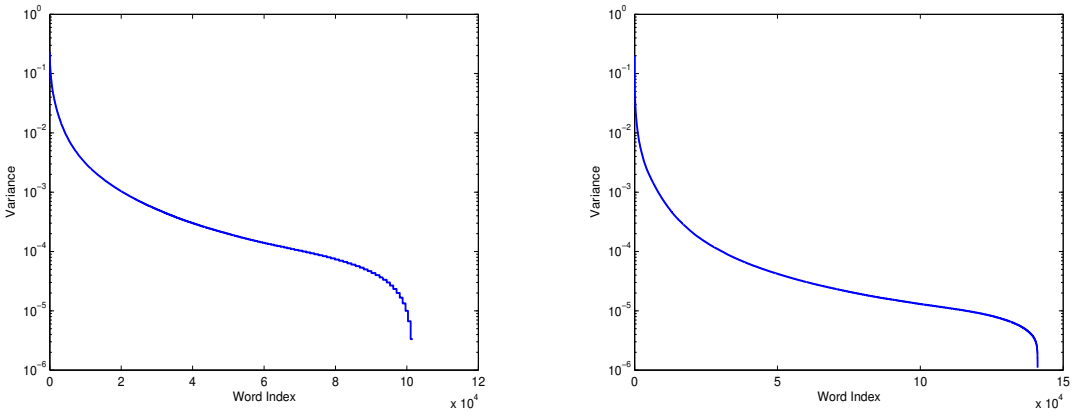

Figure 3.2: Sorted variances of 102,660 words in NYTimes (left) and 141,043 words in PubMed (right)

Table 3.1: Words associated with the top 5 sparse principal components in NYTimes

1st PC	2nd PC	3rd PC	4th PC	5th PC
million	point	official	president	school
percent	play	government	campaign	program
business	team	united_states	bush	children
company	season	u_s	administration	student
market	game	attack		
companies				

After the pre-processing steps, it takes our algorithm around 20 seconds to search for a range of λ and find one sparse principal component with the target cardinality (for the NYTimes data in our current implementation on a MacBook laptop with 2.4 GHz Intel Core 2 Duo processor and 2 GB memory).

A surprising finding is that the safe feature elimination test, combined with the fact that word variances decrease rapidly, enables our block coordinate ascent algorithm to work on covariance matrices of order at most $n = 500$, instead of the full order ($n = 102660$) covariance matrix for NYTimes, so as to find a solution with

Table 3.2: Words associated with the top 5 sparse principal components in PubMed

1st PC	2nd PC	3rd PC	4th PC	5th PC
patient	effect	human	tumor	year
cell	level	expression	mice	infection
treatment	activity	receptor	cancer	age
protein	concentration	binding	maligant	children
disease	rat		carcinoma	child

cardinality of around 5. In the case of PubMed, our algorithm only needs to work on covariance matrices of order at most $n = 1000$, instead of the full order ($n = 141,043$) covariance matrix. Thus, at values of the penalty parameter λ that a target cardinality of 5 commands, we observe a dramatic reduction in problem sizes, about $150 \sim 200$ times smaller than the original sizes respectively. This motivates our conclusion that sparse PCA is in a sense, easier than PCA itself.

3.5 Summary

The safe feature elimination result, coupled with a fast block coordinate ascent algorithm, allows to solve sparse PCA problems for very large scale, real-life data sets. The overall method works especially well when the target cardinality of the result is small, which is often the case in applications where interpretability by a human is key. The algorithm we proposed has better computational complexity, and in practice converges much faster than, the first-order algorithm developed in d'Aspremont et al. [9] . Our experiments on text data also hint that the sparse PCA can be a promising approach towards summarizing and organizing a large text corpus, which will be further investigated in Chapter 4.

Chapter 4

Applications

4.1 Introduction

As mentioned in Chapter 1, one major motivation for Sparse PCA theory and algorithm research is that the sparse principal components found by such algorithms should be more interpretable than their classic counterparts. To validate that motivation, we apply Sparse PCA to real data sets of various nature, illustrating the interpretability of sparse principal components in each case.

Then, we focus on the interesting application of Sparse PCA to analysis of text data, where each sparse principal component is found to correspond to some coherent aspect of the text data. Typically, the few (say 5 to 10) words in each sparse principal component can be thought to represent a particular concept or topic. These findings suggest that Sparse PCA can be used as an attractive alternative for topic modelling.

4.2 Non-text data

4.2.1 Senate voting data

In this section, we analyze the voting records of the 109th US Senate (2000-2004) There were 101 senators (one extra Senator is due to a replacement during the term) and 48 bills involved. To simplify, the votes are divided into yes (coded as 1) or no (coded as -1), and other votes are coded as 0.

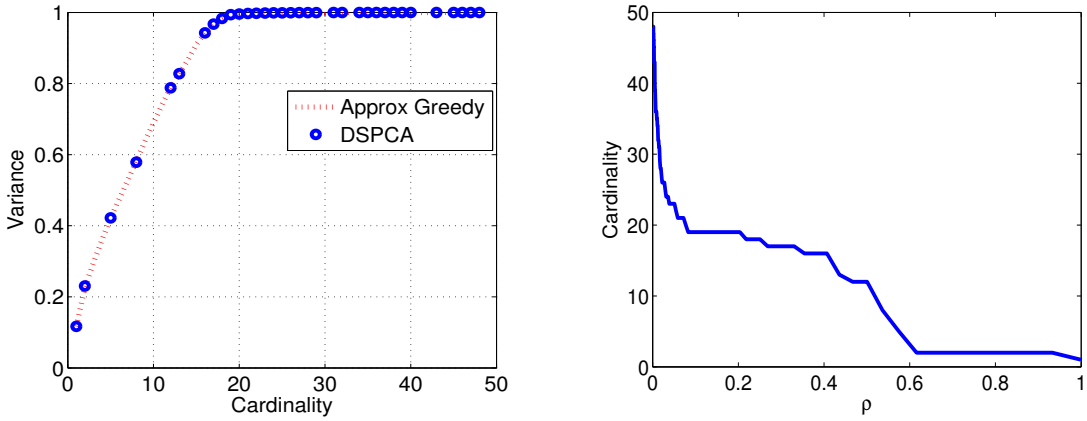

Figure 4.1: *Left:* Explained variance as a function of cardinality. *Right:* Cardinality as a function of penalty parameter ρ.

Each senator's voting record can be viewed as a point in a 48-dimensional space. By applying PCA, and projecting each senator's voting record onto a two-dimensional subspace of maximum variance, we can see that senators are almost perfectly separated by partisanship (Fig. 4.2). However, since the principal components involve all the bills, it is hard to tell which bills are most responsible for the explained variance. By applying Sparse PCA to the voting record, we aim to find a few bills that not only divide the senators according to partisanship, but also reveal which topics are most controversial within the Republican and Democratic parties. Fig. 4.2 (right)

shows the senators' voting records, projected onto the first two sparse principal components. We note that in the two-dimensional space senators are still divided by partisanship. In fact, many republican senators perfectly coincide with each other and so are democratic senators. In contrast to Fig. 4.2 (left), the cardinalities associated with the first and second sparse principal components are 5 and 2 respectively, which makes it possible to interpret the coordinates.

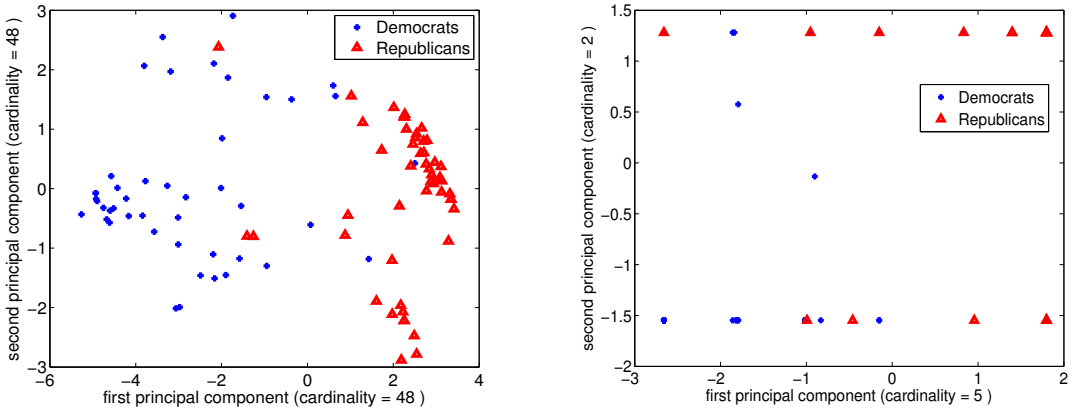

Figure 4.2: 109th Senate's voting record projected onto the top 2 principal components.

Let us examine the bills appearing in the first two sparse principal components. For the first sparse PC, the corresponding bills' brief description is as follows:

- S. 1932, As Amended; Deficit Reduction Act of 2005.
- S. Con. Res. 83; An original concurrent resolution setting forth the congressional budget for the United States Government for fiscal year 2007 and including the appropriate budgetary levels for fiscal years 2006 and 2008 through 2011.
- S. 3930, As Amended; Military Commissions Act of 2006.
- S. 403, As Amended; Child Interstate Abortion Notification Act.

- Passage of S. 397, As Amended; Protection of Lawful Commerce in Arms Act.

The brief description for the two bills in the second sparse principal component are:

- H. R. 3045; Dominican Republic-Central America-United States Free Trade Agreement Implementation Act.
- S. 1307; Dominican Republic-Central America-United States Free Trade Agreement Implementation Act.

A glance at these bills tells us that the major controversial issues between Democrats and Republicans are topics such as "abortion", "military", "budget", and "free trade".

Fig 4.1 plots the variance explained by the first sparse principal component divided by that explained by the first PC, as a function of the cardinality of the sparse PC. Fig. 4.1 also shows how the cardinality of the first sparse PC varies as the penalty parameter ρ is changed in the DSPCA code.We can see that when 19 out of 48 variables (bills) are used, sparse PCA almost achieves the same statistical fidelity as standard PCA does.

4.2.2 Stock market data

In this section, we investigate the historical prices of S&P500 stocks over 5 years, from June 1st, 2005, through June 1st, 2010. By taking out the stocks with less than 5 years of history, we end up with 472 stocks, each having daily closing prices over 1259 trading days. The prices are first adjusted for dividends and splits and then used to calculate daily log returns. Each day's return can be represented as a point in $\mathbf{R}^{472}$.

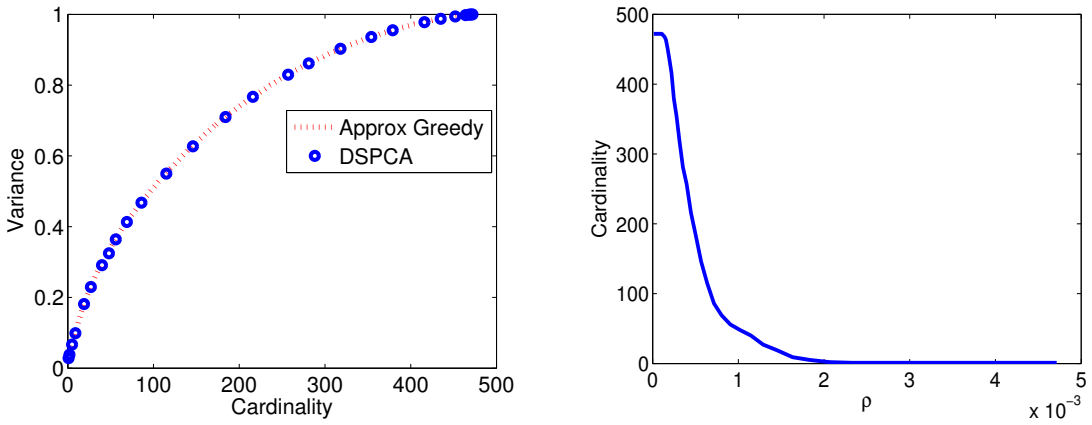

Figure 4.3: *Left:* Explained variance as a function of cardinality. *Right:* Cardinality as a function of penalty parameter ρ.

Fig. 4.3 shows the explained variance as a function of 1st PC's cardinality. It seems hard to say that the 1st PC is sparse, since there is no natural "kink" in that curve. That is, we need almost 300 out of the total 472 stocks to explain at least 90% of the variance explained by the 1st PC from PCA. However, when we inspect the sparse PCs with increasing cardinalities, we note that initially only stocks from the "Financials" sector come to play and later until, at cardinality 32, do we see companies from other sectors appearing in the 1st sparse PC. So we take the first sparse PC with cardinality equal to 32. Then we solve for the 2nd sparse PC, and using the same guideline to arrive at a cardinality of 26.

Figure 4.4 show the stock returns projected onto the 2-dimensional subspaces spanned by the top 2 PCs and top 2 sparse PCs, respectively. Comparing these two plots, we observe two interesting phenomena:

- Although the top 2 principal components from PCA explain more variance (as seen from the larger range of the axes in the left over the right panel), the two sparse principal components from DSPCA involve only 58 out of 472 stocks (32

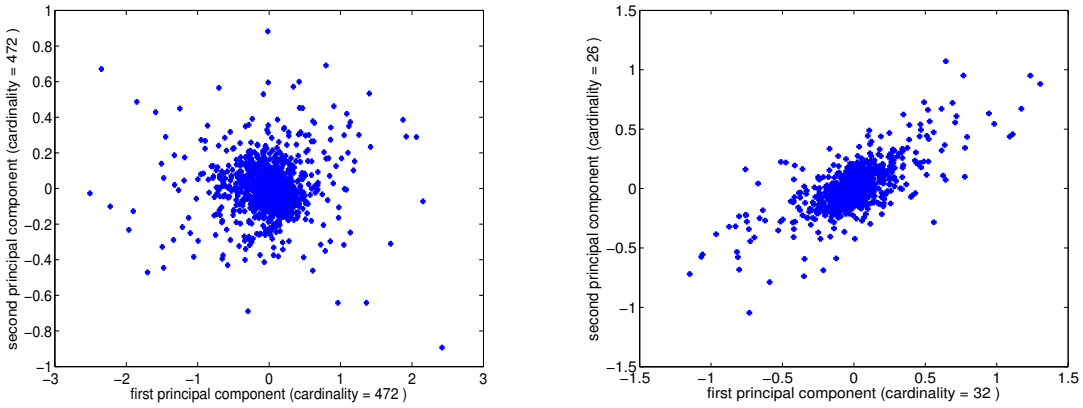

Figure 4.4: S&P500 daily returns projected onto the top 2 principal components. For PCA (left) and sparse PCA (right).

on the first PC and another distinct 26 on the second). Furthermore, 31 of the 32 stocks in the first sparse PC are all from the sector "Financials", and that almost all 26 stocks in the second sparse PC come from "Energy" and "Materials" except 2 from "Financials" and 1 from "Information Technology", as shown in Table 4.1. Considering that there are 10 sectors in total, this is quite interesting as Sparse PCA is able to identify the right groups (industry factors) that explains most of the variance. Our data covers June 2005 through June 2010 where a severe financial crisis took place, and the key role of the Financial sector is revealed purely through our sparse PCA analysis.

- In Fig. 4.4 (left), the projected data appears symmetrically distributed around its center. In contrast, In Fig. 4.4 (right), we observe a definite orientation. Since the horizontal axis (first PC) corresponds to "Financials" and the vertical one to "Energy" and "Materials", the sparse PCA analysis tells us that these two sectors are positively correlated.

1st PC	2nd PC
GNW (Genworth Financial)	CHK (Chesapeake Energy)
HIG (Hartford Financial)	AKS (AK Street Holding Corp.)
LNC (Lincoln National)	CLF (Cliffs Natural Resources)
PFG (Principal Financial)	X (United States Steel Corp.)
PRU (Prudential Financial)	MEE (Massey Energy Company)
XL (XL Capital)	CNX (CONSOL Energy Inc.)
C (Citigroup)	BTU (Peabody Energy)
BAC (Bank of America)	COG (Cabot Oil & Gas)
HBAN (Huntington Bancshares)	FCX (Freeport-McMoran Cp & Gld)
FITB (Fifth Third Bancorp)	TIE (Titanium Metals Corp)
RF (Regions Financial)	ATI (Allegheny Technologies Inc.)
MI (Marshall & Ilsley)	DNR (Denbury Resources Inc.)
KEY (KeyCorp)	NOV (National Oilwell Varco Inc.)
STI (SunTrust Banks)	SWN (Southwestern Energy)
ZION (Zions Bancorp)	RDC (Rowan Cos.)
CBG (CB Richard Ellis Group)	AIG (American International Group)
PLD (ProLogis)	SII (Smith International)
AIG (American International Group)	PXD (Pioneer Natural Resources)
COF (Capital One Financial)	HP (Helmerich & Payne)
WFC (Wells Fargo)	GNW (Genworth Financial Inc.)
MS (Morgan Stanley)	NBR (Nabors Industries)
ETFC (E-Trade)	PLD (ProLogis)
JNS (Janus Capital Group)	AA (Alcoa)
HST (Host Hotels & Resorts)	HES (Hess Corp)
LEN (Lennar Corp.)	CBG (CB Richard Ellis Group)
STT (State Street Corp.)	NUE (Nucor Corp.)
SLM (SLM Corp.)	
MET (MetLife Inc.)	
FHN (First Horizon National)	
AIV (AIMCO)	
KIM (Kimco Realty)	
CMA (Comeria)	

Table 4.1: Top 2 PCs from DSPCA

4.3 Text data

A text corpus can be represented as a matrix $X = \{x_{ij}\}$, with each row $i = 1, \ldots, m$ corresponding to a particular document, each column $j = 1, \ldots, n$

corresponding to a particular text token, and each element encoding the number of times each token appears in each document, resulting in a large, sparse matrix. This encoding of value x_{ij} is open to a variety of methods. One could use a straight count of the number of times token j appeared in document i, or a binary matrix where x_{ij} is a 0/1 indicator of which tokens appeared in which documents. The TF-IDF approach(Salton [37]) is a popular representation for many text processing applications, and the effects of several other candidate representations are tested in Nakov et al. [29]. It's important to remember that choosing this encoding is an important first step which can greatly impact our results.

4.3.1 20-newsgroups data

Our first text data set is a small version of the "20-newsgroups" data[1]. The data records binary occurrences of 100 specific words across 16242 postings, where the postings have been tagged by the highest level domain in Usenet.

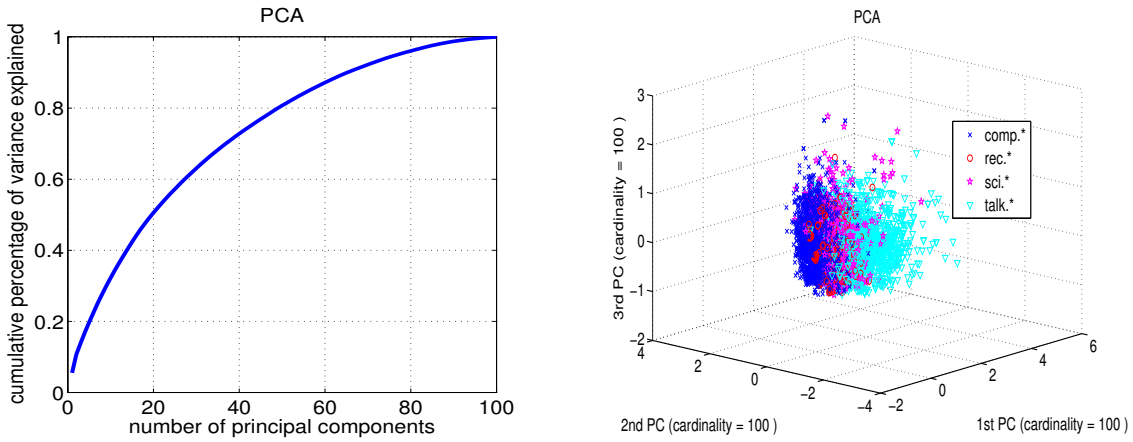

Figure 4.5: PCA with 20-Newsgroups data. *Left:* Explained variance vs. number of PCs. *Right:* 3D visualization via PCA.

[1] available from `http://cs.nyu.edu/~roweis/data.html`.

Each posting is viewed as one point in a 100-dimensional space. We begin with a standard PCA on the data. Fig. 4.5 (left) shows the cumulative percentage of variance explained as we increase the number of principal components. The slow increase means that the data does not lie within a subspace of significantly low dimension. We can anyway proceed to visualize the data: Fig. 4.5 (right) is the result obtained by projecting it on a subspace of dimension 3, chosen by selecting the eigenvectors corresponding to the three largest eigenvalues of the covariance matrix. Since these 3 vectors are dense, the axes in Fig. 4.5 (right) do not have a clear interpretation.

With Sparse PCA, we hope to find a set of corresponding sparse principal components, which still help with visualization nearly as well as PCA does, and yet reveal some interesting structure. To achieve this, we have run the block coordinate ascent algorithm developed in Chapter 3 on the data with a range of values for the penalty parameter ρ. We obtained a plot of the variance explained by the first sparse principal component (PC), as a function of its cardinality (Fig. 4.6). We then selected a cardinality that can explain at least 90% of the variance explained by the the first principal component obtained from PCA. Then we have deflated the covariance matrix by taking out the part due to the first sparse PC, and then repeated the above procedure to obtain the second sparse PC. In the same way, we have solved for the third sparse PC. Fig. 4.6 also shows the projection of the data on the 3-dimensional subspace that is spanned by the three sparse PCs obtained above.

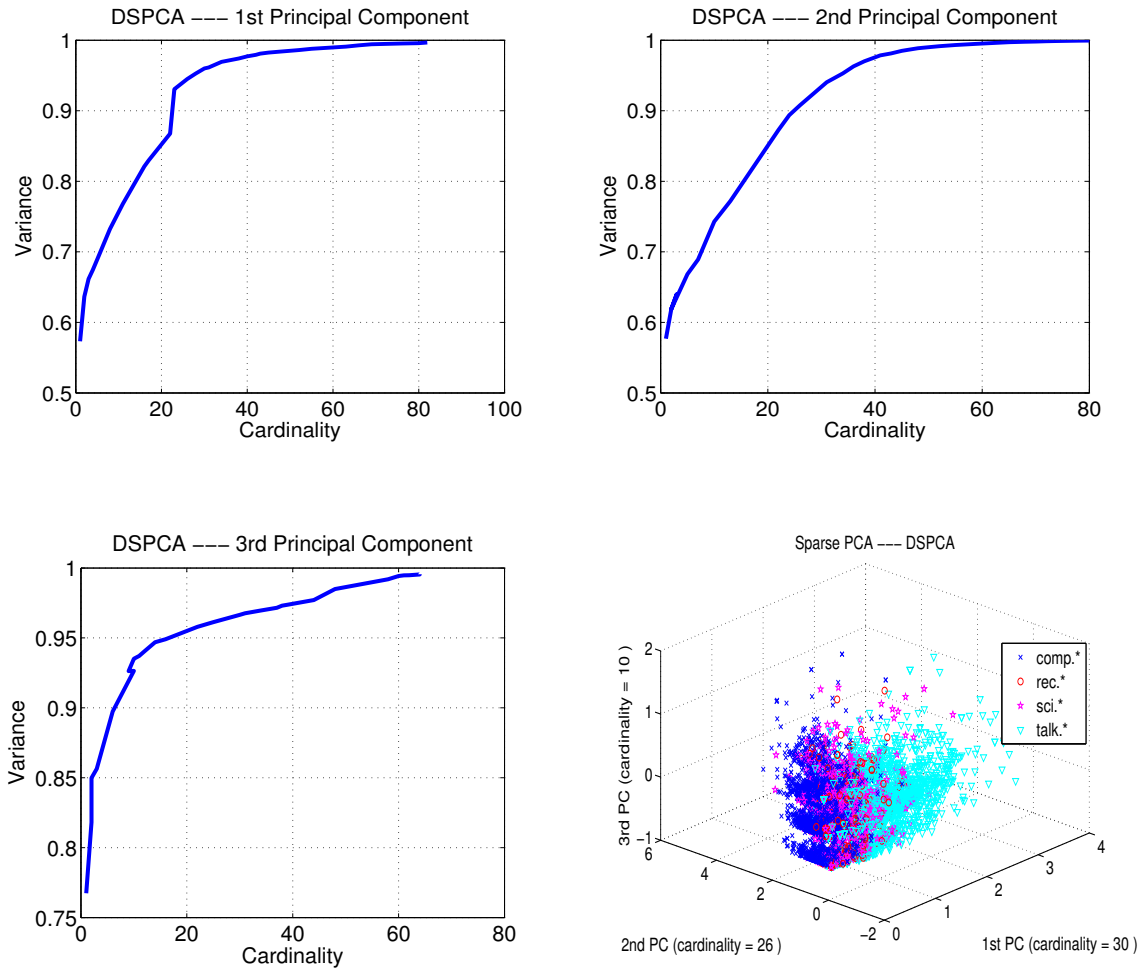

Figure 4.6: Sparse PCA on the 20Newsgroups data set. First three principal components and 3D visualization. The first three principal components have cardinalities 26, 30 and 10 respectively.

We first note that only a small number of words, out of the total of 100 words that can appear in each sparse PC, can explain more than 90% of variance explained by the corresponding PC. Specifically, we obtain 30 words for the first PC, 26 for the second, and 10 for the third. The lists of words associated with each sparse PCs is given in Table 4.2, and reveals some structure about each one of the sparse PCs. That is, the 30 words associated with the first sparse PC are almost all about politics and religion, the 26 words in the second sparse PC are all computer-related, and the

majority of the 10 words in the third sparse PC concerns science. Hence, applying sparse PCA to this data set allows to discover structure that is otherwise hidden in the standard PCA, for example that the first principal component is mainly related to politics and religion.

1st PC (30 words)	2nd PC (26 words)	3rd PC (10 words)
fact	help	problem
question	problem	university
world	system	email
course	email	state
case	windows	research
problem	program	science
god	computer	phone
government	software	world
human	university	fact
state	version	question
number	files	
christian	drive	
evidence	data	
law	card	
power	dos	
religion	god	
children	disk	
jesus	pc	
system	graphics	
rights	ftp	
war	memory	
jews	christian	
help	phone	
bible	video	
earth	fact	
science	display	
research		
israel		
president		
gun		

Table 4.2: Words associated with the first three sparse PCs.

4.3.2 New York Times data

Our second data set is the collection of 1,288 news articles published in 2009 by the *New York Times*'s International section mentioning the word "China". We tokenize the articles by unigrams, remove no stop words, and perform no stemming. The data encodes the binary $\{0, 1\}$ values (corresponding to appearance/non-appearance) of 86,500 unique tokens. Compared to the "20 Newsgroup" data, this is a fairly large "unlabeled" data in terms of feature size (i.e. number of unique tokens). For the purpose of comparison, we run both the thresholded PCA and DSPCA algorithms over this set of text news.

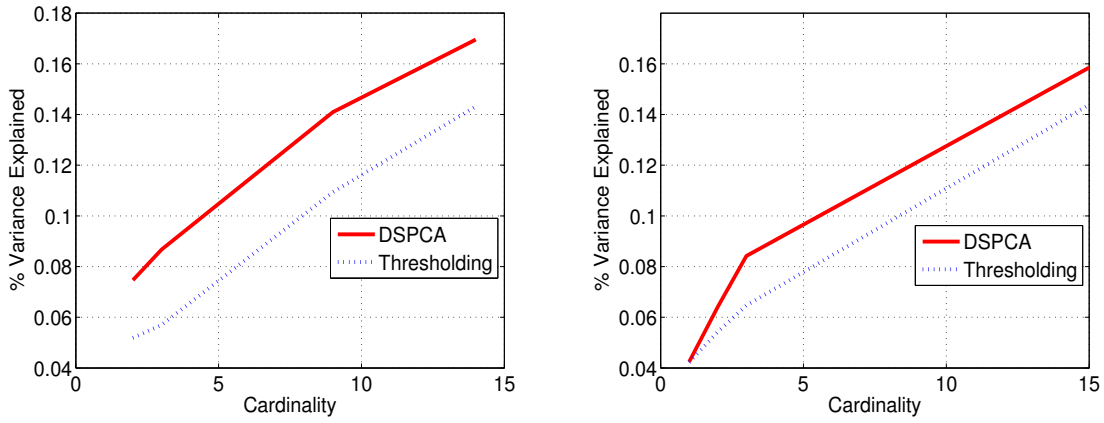

Figure 4.7: Sparse PCA on 1,288 *New York Times* articles mentioning the word "China".

Figure 4.7 shows the percentage of explained variance as a function of cardinality. Here we see DSPCA does outperform Thresholded PCA, though not by a big margin. Although we do not have ground truth, Table 4.3 and Table 4.4 contains words selected by two algorithms respectively as we increase cardinality. Words selected by DSPCA appear much more meaningful than those chosen by thresholded PCA at the same cardinality.

$k = 2$	$k = 3$	$k = 9$	$k = 14$
united	american	washington	international
states	united	american	would
	states	administration	will
		united	washington
		states	american
		president	administration
		obama	united
		countries	states
		nations	president
			obama
			counties
			nations
			policy
			nuclear

Table 4.3: 1st PC from DSPCA on 1,288 *New York Times* articles mentioning the word "China" for various values of the eigenvector cardinality k.

$k = 2$	$k = 3$	$k = 9$	$k = 14$
even	even	even	would
like	like	we	new
	states	like	even
		now	we
		this	like
		will	now
		united	this
		states	will
		if	united
			states
			world
			so
			some
			if

Table 4.4: 1st PC from Thresholded PCA on 1,288 *New York Times* articles mentioning the word "China" for various values of the eigenvector cardinality k.

Table 4.5 lists the words corresponding to the top 6 sparse principal components.

This tells us that among all these articles mentioning the word "China", there are a few important topics running through the text collection: the first sparse principal component probably represents China-US relationships, the third sparse principal component centers around world economy and financial crisis that was happening in 2009, the fourth sparse principal component is related to geographic politics that is important to China, and the fifth sparse principal component is purely about China's biggest state-run news agency. It is also very interesting to observe that all the stop words are grouped into one sparse principal component (the second one), instead of polluting the other sparse principal component. Finally, it is a bit surprising that the sixth sparse principal component essentially is a set of synonyms.

1st PC	2nd PC	3rdPC	4th PC	5th PC	6th PC
united	all	world	chinese	news	his
states	if	economic	beijing	official	i
president	most	global	chinas	reported	mr
obama	no	percent	north	agency	he
american	many	economy	korea	xinhua	him
washington	we	billion	russia		
countries	so	financial	iran		
nations	like	crisis	program		
admin.	what	growth	nuclear		
	now		weapons		
	only		sanctions		
	there				

Table 4.5: Words associated with the top 6 sparse principal components

We also run the Sparse PCA analysis on the news articles published by the *New York Times*'s International section mentioning other target words such as "Russia" and "Germany". The findings are very similar, so we don't report them here to avoid redundancy.

For a larger set of *New York Times*'s articles instead of a small subset mentioning

1st PC	2nd PC	3rd PC	4th PC	5th PC
million	point	official	president	school
percent	play	government	campaign	program
business	team	united_states	bush	children
company	season	u_s	administration	student
market	game	attack		
companies				

Table 4.6: Words associated with the top 5 sparse principal components in NY-Times

a particular word, we already report our findings in Section 3.4 of Chapter 3, where the main purpose is to demonstrate that our newly proposed algorithm can be safely applied to large data sets. We repeat the results here to emphasize that Sparse PCA do uncover interesting topics buried in a large unlabeled text corpus. That set of NYTimes news articles is much larger, containing $300,000$ articles and has a dictionary of $102,660$ unique words after stop words have been removed. The top 5 sparse principal components are shown in Table 4.6. Clearly the first principal component is about business, the second one about sports, the third about U.S., the fourth about politics and the fifth about education. Bear in mind that the NYTimes data from UCI Machine Learning Repository "have no class labels, and for copyright reasons no filenames or other document-level metadata" (Frank and Asuncion [14]). The sparse principal components still unambiguously identify and perfectly correspond to the topics used by *The New York Times* itself to classify articles on its own website.

4.4 Summary

In this chapter, we perform the Sparse PCA analysis on data sets ranging from senate voting, stock returns, 20-newsgroup to *New York Times*'s articles. In all cases, we found that the results from Sparse PCA are much more interpretable and hence

usually reveal some interesting patterns buried in the unorganized data. The findings in text data are particularly interesting, as this suggest the possibility of using Sparse PCA to better organize or to better present a large text collection. For example, Sparse PCA might be run on top of the search results returned by Google, so as to better direct users towards the subset regarding a particular topic or aspect in the query results.

Chapter 5

Conclusions

In this dissertation, we first discuss several formulations for Sparse PCA, as well as the algorithms for solving the formulations and a few greedy methods. We empirically show that the particular formulation based on semidefinite relaxation called DSPCA typically has the best performance, especially when only limited number of samples are available or when the data is noisy.

We then develop a block coordinate ascent algorithm for solving DSPCA with better dependence on problem size. We also explicitly leverage a safe feature elimination procedure to make our code more scalable. We show that our algorithm converges much faster than the existing first-order algorithm in practice and demonstrate that our code can handle huge real data sets. This newly developed package should make it easier for applying Sparse PCA (using DSPCA) to large data sets typically found in contemporary applications.

We also demonstrate that Sparse PCA does bring more interpretability and hence gives rise to new interesting findings in various real data sets. In particular, we found that the extremely sparse principal components in text news usually correspond to

some semantically coherent concepts or topics. This opens the door to potentially using Sparse PCA as an attractive alternative approach to topic models.

There are still many questions remained to be answered in terms of Sparse PCA research. First, outside of the (locally) convergent algorithm in Journée et al. [20], very few methods handle the problem of simultaneously finding several leading sparse principal components. Also, most methods (even extremely simple ones) perform well enough on easy, "natural" data sets while only the most expensive semidefinite relaxations seem to produce good bounds on the random matrices used in compressed sensing applications, or when only a few samples are available for example. Characterizing what makes "natural" data sets easier than random ones remains an open problem at this point. It is also not clear yet how to extend the statistical optimality statements of Amini and Wainwright [2] to broader (e.g. deterministic) classes of matrices.

Bibliography

[1] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5:13–51, 1995.

[2] A.A. Amini and M. Wainwright. High-dimensional analysis of semidefinite relaxations for sparse principal components. *The Annals of Statistics*, 37(5B): 2877–2921, 2009.

[3] T. W. Anderson. *An Introduction to Multivariate Statistical Analysis*. Wiley, New York, NY, 1984.

[4] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[5] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787.

[6] J. Cadima and I. T. Jolliffe. Loadings and correlations in the interpretation of principal components. *Journal of Applied Statistics*, 22:203–214, 1995.

[7] E. J. Candès and T. Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005.

[8] A. d'Aspremont, L. El Ghaoui, M. Jordan, and G. Lanckriet. A direct formulation of sparse PCA using semidefinite programming. *SIAM Review*, 49(3), 2007.

[9] A. d'Aspremont, L. El Ghaoui, M.I. Jordan, and G. R. G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. *SIAM Review*, 49 (3):434–448, 2007.

[10] A. d'Aspremont, F. Bach, and L. El Ghaoui. Optimal solutions for sparse principal component analysis. *Journal of Machine Learning Research*, 9:1269–1294, 2008.

[11] D. L. Donoho and J. Tanner. Sparse nonnegative solutions of underdetermined linear equations by linear programming. *Proc. of the National Academy of Sciences*, 102(27):9446–9451, 2005.

[12] L. El Ghaoui. On the quality of a semidefinite programming bound for sparse principal component analysis. arXiv:math/060144, February 2006.

[13] M. Fazel, H. Hindi, and S. Boyd. A rank minimization heuristic with application to minimum order system approximation. *Proceedings American Control Conference*, 6:4734–4739, 2001.

[14] A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

[15] G.H. Golub and C.F. Van Loan. Matrix computation. *North Oxford Academic*, 1990.

[16] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[17] I. T. Jolliffe. Rotation of principal components: choice of normalization constraints. *Journal of Applied Statistics*, 22:29–35, 1995.

[18] I. T. Jolliffe. *Principal Component Analysis*. Springer, New York, NY, 2004.

[19] I. T. Jolliffe, N.T. Trendafilov, and M. Uddin. A modified principal component technique based on the LASSO. *Journal of Computational and Graphical Statistics*, 12:531–547, 2003.

[20] M. Journée, Y. Nesterov, P. Richtárik, and R. Sepulchre. Generalized power method for sparse principal component analysis. *arXiv:0811.4724*, 2008.

[21] H.F. Kaiser. The varimax criterion for analytic rotation in factor analysis. *Psychometrika*, 23(3):187–200, 1958.

[22] C. Lemaréchal and F. Oustry. Semidefinite relaxations and Lagrangian duality with application to combinatorial optimization. *INRIA, Rapport de recherche*, 3710, 1999.

[23] L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization*, 1(2):166–190, 1991.

[24] L. Mackey. Deflation methods for sparse pca. *Advances in Neural Information Processing Systems*, 21:1017–1024, 2009.

[25] B. Moghaddam, Y. Weiss, and S. Avidan. Generalized spectral bounds for sparse LDA. In *International Conference on Machine Learning*, 2006.

[26] B. Moghaddam, Y. Weiss, and S. Avidan. Spectral bounds for sparse PCA: exact and greedy algorithms. *Advances in Neural Information Processing Systems*, 18, 2006.

[27] B. Moghaddam, Y. Weiss, and S. Avidan. Fast Pixel/Part Selection with Sparse Eigenvectors. *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, 2007.

[28] C. Moler and C. Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49, 2003.

[29] Preslav Nakov, Antonia Popova, and Plamen Mateev. Weight functions impact on LSA performance. In *EuroConference RANLP'2001 (Recent Advances in NLP*, pages 187–193, 2001.

[30] B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM J. Comput.*, 24(2):227–234, 1995.

[31] Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

[32] Y. Nesterov. *Introductory Lectures on Convex Optimization*. Springer, 2003.

[33] Y. Nesterov. Smoothing technique and its applications in semidefinite optimization. *Mathematical Programming*, 110(2):245–259, 2007.

[34] JO Neuhaus and C. Wrigley. The quartimax method: an analytical approach to orthogonal simple structure. *British Journal of Statistical Psychology*, 7:81–91, 1954.

[35] O.Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *Journal of Machine Learning Research*, 9:485–516, March 2008.

[36] Y. Saad. *Numerical methods for large eigenvalue problems*. Manchester Univ Press, 1992. URL `http://www-users.cs.umn.edu/$\sim$saad/books.html`.

[37] G. Salton. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[38] Haipeng Shen and Jianhua Z. Huang. Sparse principal component analysis via regularized low rank matrix approximation. *J. Multivar. Anal.*, 99:1015–1034, July 2008. ISSN 0047-259X. doi: 10.1016/j.jmva.2007.06.007. URL `http://portal.acm.org/citation.cfm?id=1370298.1370362`.

[39] B.K. Sriperumbudur, D.A. Torres, and G.R.G. Lanckriet. Sparse eigen methods by DC programming. *Proceedings of the 24th international conference on Machine learning*, pages 831–838, 2007.

[40] R. Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal statistical society, series B*, 58(1):267–288, 1996.

[41] Zaiwen Wen, Donald Goldfarb, Shiqian Ma, and Katya Scheinberg. Row by row methods for semidefinite programming. Technical report, Dept of IEOR, Columbia University, 2009.

[42] Y. Zhang, A. d'Aspremont, and L. El Ghaoui. Sparse PCA: Convex relaxations, algorithms and applications. In M. Anjos and J.B. Lasserre, editors, *Handbook on Semidenite, Cone and Polynomial Optimization: Theory, Algorithms, Software and Applications*. Springer, 2011. To appear.

[43] Youwei Zhang and Laurent El Ghaoui. Large-scale sparse principal component analysis with application to text data. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[44] Z. Zhang, H. Zha, and H. Simon. Low rank approximations with sparse factors I: basic algorithms and error analysis. *SIAM journal on matrix analysis and its applications*, 23(3):706–727, 2002.

[45] H. Zou, T. Hastie, and R. Tibshirani. Sparse Principal Component Analysis. *Journal of Computational & Graphical Statistics*, 15(2):265–286, 2006.