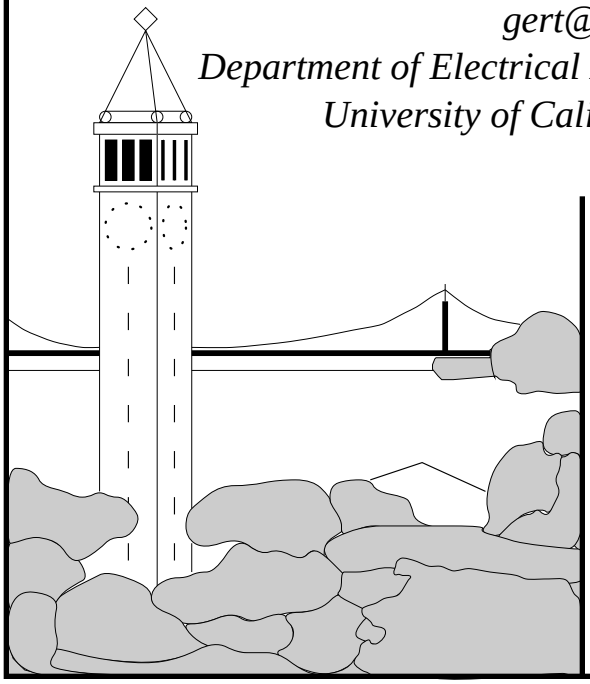
A direct formulation for sparse PCA using semidefinite programming

Alexandre d'Aspremont
alexandre.daspremont@m4x.org
Department of Electrical Engineering and Computer Science
University of California, Berkeley, CA 94720

Laurent El Ghaoui
elghaoui@eecs.berkeley.edu
Department of Electrical Engineering and Computer Science
University of California, Berkeley, CA 94720

Michael I. Jordan
jordan@cs.berkeley.edu
Division of Computer Science & Department of Statistics,
University of California, Berkeley, CA 94720

Gert R.G. Lanckriet
gert@cs.berkeley.edu
Department of Electrical Engineering and Computer Science
University of California, Berkeley, CA 94720



Report No. UCB/CSD-04-1330

June, 2004

Computer Science Division (EECS)
University of California
Berkeley, California 94720

A direct formulation for sparse PCA using semidefinite programming

Alexandre d'Aspremont
alexandre.daspremont@m4x.org
Department of Electrical Engineering and Computer Science
University of California, Berkeley, CA 94720

Laurent El Ghaoui
elghaoui@eecs.berkeley.edu
Department of Electrical Engineering and Computer Science
University of California, Berkeley, CA 94720

Michael I. Jordan
jordan@cs.berkeley.edu
Division of Computer Science & Department of Statistics,
University of California, Berkeley, CA 94720

Gert R.G. Lanckriet
gert@cs.berkeley.edu
Department of Electrical Engineering and Computer Science
University of California, Berkeley, CA 94720

June, 2004

Abstract

We examine the problem of approximating, in the Frobenius-norm sense, a positive, semidefinite symmetric matrix by a rank-one matrix, with an upper bound on the cardinality of its eigenvector. The problem arises in the decomposition of a covariance matrix into sparse factors, and has wide applications ranging from biology to finance. We use a modification of the classical variational representation of the largest eigenvalue of a symmetric matrix, where cardinality is constrained, and derive a semidefinite programming based relaxation for our problem. We also discuss Nesterov's smooth minimization technique applied to the SDP arising in the direct sparse PCA method. The method has complexity $O(n^4\sqrt{\log(n)}/\epsilon)$, where n is the size of the underlying covariance matrix, and ϵ is the desired absolute accuracy on the optimal value of the problem.

1 Introduction

Principal component analysis (PCA) is a popular tool for data analysis and dimensionality reduction. It has applications throughout science and engineering. In essence, PCA finds linear combinations of the

variables (the so-called principal components) that correspond to directions of maximal variance in the data. It can be performed via a singular value decomposition (SVD) of the data matrix A, or via an eigenvalue decomposition if A is a covariance matrix.

The importance of PCA is due to several factors. First, by capturing directions of maximum variance in the data, the principal components offer a way to compress the data with minimum information loss. Second, the principal components are uncorrelated, which can aid with interpretation or subsequent statistical analysis. On the other hand, PCA has a number of well-documented disadvantages as well. A particular disadvantage that is our focus here is the fact that the principal components are usually linear combinations of *all* variables. That is, all weights in the linear combination (known as *loadings*), are typically non-zero. In many applications, however, the coordinate axes have a physical interpretation; in biology for example, each axis might correspond to a specific gene. In these cases, the interpretation of the principal components would be facilitated if these components involve very few non-zero loadings (coordinates). Moreover, in certain applications, e.g., financial asset trading strategies based on principal component techniques, the sparsity of the loadings has important consequences, since fewer non-zero loadings imply fewer fixed transaction costs.

It would thus be of interest to be able to discover "sparse principal components", i.e., sets of sparse vectors spanning a low-dimensional space that explain most of the variance present in the data. To achieve this, it is necessary to sacrifice some of the explained variance and the orthogonality of the principal components, albeit hopefully not too much.

Rotation techniques are often used to improve interpretation of the standard principal components [5]. [14] considered simple principal components by restricting the loadings to take values from a small set of allowable integers, such as 0, 1, and -1. [3] propose an ad hoc way to deal with the problem, where the loadings with small absolute value are thresholded to zero. We will call this approach "simple thresholding." Later, a method called SCoTLASS was introduced by [6] to find modified principal components with possible zero loadings. In [15] a new approach, called sparse PCA (SPCA), was proposed to find modified components with zero loadings, based on the fact that PCA can be written as a regression-type optimization problem. This allows the application of LASSO [11], a penalization technique based on the L_1 norm.

In this paper, we propose a direct approach (called DSPCA in what follows) that improves the sparsity of the principal components by directly incorporating a sparsity criterion in the PCA problem formulation and then relaxing the resulting optimization problem, yielding a convex optimization problem. In particular, we obtain a convex semidefinite programming (SDP) formulation.

SDP problems can be solved in polynomial time via general-purpose interior-point methods [10, 12], and our current implementation of DSPCA makes use of these general-purpose methods. This suffices for an initial empirical study of the properties of DSPCA and for comparison to the algorithms discussed above on problems of small to medium dimensionality. For high-dimensional problems, the general-purpose methods are not viable and it is necessary to attempt to exploit special structure in the problem. It turns out that our problem can be expressed as a special type of saddle-point problem that is well suited to recent specialized algorithms, such as those described in [9, 8, 1]. These algorithms offer a significant reduction in computational time compared to generic SDP solvers. In the current paper, however, we restrict ourselves to an investigation of the basic properties of DSPCA on problems for which the generic methods are adequate.

Our paper is structured as follows. In Section 2, we show how to efficiently derive a sparse rank-one approximation of a given matrix using a semidefinite relaxation of the sparse PCA problem, and briefly explain how to generalize the approach to non-square matrices. In Section 3, we derive an interesting robustness interpretation of our technique, and in Section 4 we describe how to use this interpretation in order to decompose a matrix into sparse factors. Section 5 outlines different algorithms that can be used

to solve the problem, while Section 6 presents numerical experiments comparing our method with existing techniques.

Notation

In this paper, $\mathbf{S}^n$ is the set of symmetric matrices of size n, and Δ_n the corresponding spectahedron (set of positive semi-definite matrices with unit trace). We denote by $\mathbf{1}$ a vector of ones (with size inferred from context), while $\mathbf{Card}(x)$ denotes the cardinality (number of non-zero elements) of a vector x. For $X \in \mathbf{S}^n$, we denote by $\|X\|_F$ is the Frobenius norm of X, i.e., $\|X\|_F = \sqrt{\mathbf{Tr}(X^2)}$, by $\lambda^{\max}(X)$ the maximum eigenvalue of X and by $\|X\|_\infty = \max_{1\leq i,j\leq n} |X_{ij}|$, while $|X|$ is the matrix whose elements are the absolute values of the elements of X.

2 Sparse eigenvectors

In this section, we derive a semidefinite programming (SDP) relaxation for the problem of approximating a symmetric matrix by a rank one matrix with an upper bound on the cardinality of its eigenvector. We first reformulate this as a variational problem, we then obtain a lower bound on its optimal value via an SDP relaxation (we refer the reader to [13] or [2] for an overview of semidefinite programming).

2.1 Single factor, A positive semidefinite

Let $A \in \mathbf{S}^n$ be a given $n \times n$ positive semidefinite, symmetric matrix and k be an integer with $1 \leq k \leq n$. We consider the problem:

$$\begin{array}{rll} \Phi_k(A) := & \min & \|A - xx^T\|_F \\ & \text{subject to} & \mathbf{Card}(x) \leq k, \end{array} \tag{1}$$

in the variable $x \in \mathbf{R}^n$. We can solve instead the following equivalent problem:

$$\begin{array}{rll} \Phi_k^2(A) = & \min & \|A - \lambda xx^T\|_F^2 \\ & \text{subject to} & \|x\|_2 = 1, \;\; \lambda \geq 0, \\ & & \mathbf{Card}(x) \leq k, \end{array}$$

in the variable $x \in \mathbf{R}^n$ and $\lambda \in \mathbf{R}$. Minimizing over λ, we obtain:

$$\Phi_k^2(A) = \|A\|_F^2 - \nu_k(A),$$

where

$$\begin{array}{rll} \nu_k(A) := & \max & x^T A x \\ & \text{subject to} & \|x\|_2 = 1 \\ & & \mathbf{Card}(x) \leq k. \end{array} \tag{2}$$

To compute a semidefinite relaxation of this program (see [7] or [2], for example), we rewrite (2) as:

$$\begin{array}{rll} \nu_k(A) := & \max & \mathbf{Tr}(AX) \\ & \text{subject to} & \mathbf{Tr}(X) = 1 \\ & & \mathbf{Card}(X) \leq k^2 \\ & & X \succeq 0, \;\; \mathbf{Rank}(X) = 1, \end{array} \tag{3}$$

in the symmetric, matrix variable $X \in \mathbf{S}^n$. Indeed, if X is a solution to the above problem, then $X \succeq 0$ and $\mathbf{Rank}(X) = 1$ means that we have $X = xx^T$, and $\mathbf{Tr}(X) = 1$ implies that $\|x\|_2 = 1$. Finally, if $X = xx^T$ then $\mathbf{Card}(X) \leq k^2$ is equivalent to $\mathbf{Card}(x) \leq k$.

Naturally, problem (3) is still non-convex and very difficult to solve, due to the rank and cardinality constraints. Since for every $u \in \mathbf{R}^p$, $\mathbf{Card}(u) = q$ implies $\|u\|_1 \leq \sqrt{q}\|u\|_2$, we can replace the non-convex constraint $\mathbf{Card}(X) \leq k^2$, by a weaker but convex one: $\mathbf{1}^T|X|\mathbf{1} \leq k$, where we have exploited the property that $\|X\|_F = \sqrt{x^T x} = 1$ when $X = xx^T$ and $\mathbf{Tr}(X) = 1$. If we also drop the rank constraint, we can form a relaxation of (3) and (2) as:

$$\begin{array}{lll} \overline{\nu}_k(A) := & \max & \mathbf{Tr}(AX) \\ & \text{subject to} & \mathbf{Tr}(X) = 1 \\ & & \mathbf{1}^T|X|\mathbf{1} \leq k \\ & & X \succeq 0, \end{array} \tag{4}$$

which is a semidefinite program (SDP) 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 $v_k(a)$ of the variational program in (2), hence it gives a lower bound on the optimal value $\Phi_k(A)$ of the original problem (1). Finally, the optimal solution X will not always be of rank one but we can truncate it and keep only its dominant eigenvector x as an approximate solution to the original problem (1). In Section 6 we show that in practice the solution X to (4) tends to have a rank very close to one, and that its dominant eigenvector is indeed sparse.

2.2 Extension to the non-square case

A similar reasoning involves a non-square $m \times n$ matrix A, and the problem

$$\begin{array}{ll} \min & \|A - uv^T\|_F \\ \text{subject to} & \mathbf{Card}(u) \leq k_1 \\ & \mathbf{Card}(v) \leq k_2, \end{array}$$

in the variables $(u, v) \in \mathbf{R}^m \times \mathbf{R}^n$ where $k_1 \leq m$, $k_2 \leq n$ are fixed. As before, we can reduce the problem to

$$\begin{array}{ll} \max & u^T A v \\ \text{subject to} & \|u\|_2 = \|v\|_2 = 1 \\ & \mathbf{Card}(u) \leq k_1, \ \mathbf{Card}(v)) \leq k_2, \end{array}$$

which can in turn be relaxed to

$$\begin{array}{ll} \max & \mathbf{Tr}(A^T X_{12}) \\ \text{subject to} & X \succeq 0, \ \mathbf{Tr}(X_{ii}) = 1 \\ & \mathbf{1}^T|X_{ii}|\mathbf{1} \leq k_i, \quad i = 1, 2 \\ & \mathbf{1}^T|X_{12}|\mathbf{1} \leq \sqrt{k_1 k_2}, \end{array}$$

in the variable $X \in \mathbf{S}^{m+n}$ with blocks X_{ij} for $i, j = 1, 2$. We can consider several variations on this, such as constraining $\mathbf{Card}(u) + \mathbf{Card}(v) = \mathbf{Card}(u, v)$.

3 A robustness interpretation

In this section, we show that problem (4) can be interpreted as a robust formulation of the maximum eigenvalue problem, with additive, component-wise uncertainty in the matrix A. We again assume A to be symmetric and positive semidefinite.

In the previous section, we considered a cardinality-constrained variational formulation of the maximum eigenvalue problem:

$$\begin{array}{lll} \nu_k(A) := & \max & x^T A x \\ & \text{subject to} & \|x\|_2 = 1 \\ & & \mathbf{Card}(x) \leq k. \end{array}$$

Here we look at a small variation where we penalize the cardinality and solve:

$$\begin{array}{ll} \max & x^T A x - \rho\, \mathbf{Card}^2(x) \\ \text{subject to} & \|x\|_2 = 1, \end{array}$$

in the variable $x \in \mathbf{R}^n$, where the parameter $\rho > 0$ controls the size of the penalty. This problem is again non-convex and very difficult to solve. As in the last section, we can form the equivalent program:

$$\begin{array}{ll} \max & \mathbf{Tr}(AX) - \rho\, \mathbf{Card}(X) \\ \text{subject to} & \mathbf{Tr}(X) = 1 \\ & X \succeq 0,\ \mathbf{Rank}(X) = 1, \end{array}$$

in the variable $X \in \mathbf{S}^n$. Again, we get a relaxation of this program by forming:

$$\begin{array}{ll} \max & \mathbf{Tr}(AX) - \rho \mathbf{1}^T |X| \mathbf{1} \\ \text{subject to} & \mathbf{Tr}(X) = 1 \\ & X \succeq 0, \end{array} \tag{5}$$

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

$$\max_{X \succeq 0, \mathbf{Tr}(X)=1} \ \min_{|U_{ij}| \leq \rho} \mathbf{Tr}(X(A+U)) \tag{6}$$

and we get a dual to (5) as:

$$\begin{array}{ll} \min & \lambda^{\max}(A+U) \\ \text{subject to} & |U_{ij}| \leq \rho, \quad i,j = 1, \ldots, n, \end{array} \tag{7}$$

which is a maximum eigenvalue problem with variable $U \in \mathbf{R}^{n \times n}$. This gives a natural robustness interpretation to the relaxation in (5): it corresponds to a worst-case maximum eigenvalue computation, with component-wise bounded noise of intensity ρ on the matrix coefficients.

Let us remark that we can easily move from the constrained formulation in (4) to the penalized form in (5). Suppose that we have solved the constrained problem (4) for a certain target cardinality k:

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

then this problem is equivalent to the penalized problem:

$$\begin{array}{ll} \text{max} & \mathbf{Tr}(AX) - \rho^\star \mathbf{1}^T |X| \mathbf{1} \\ \text{subject to} & \mathbf{Tr}(X) = 1 \\ & X \succeq 0, \end{array}$$

if we set the noise level $\rho^\star$ to be equal to the optimal Lagrange multiplier associated with the constraint $\mathbf{1}^T|X|\mathbf{1} \leq k$ in the constrained cardinality program (4). This means that we can directly compute the noise level ρ from the value of k and the dual solution to the constrained program in (4).

4 Sparse decomposition

Here, we use the results obtained in the previous two sections to describe a sparse equivalent to the PCA decomposition technique. Suppose that we start with a matrix $A_1 \in \mathbf{S}^n$, our objective is to decompose it in factors with target sparsity k. We solve the relaxed problem in (4):

$$\begin{array}{ll} \text{max} & \mathbf{Tr}(A_1 X) \\ \text{subject to} & \mathbf{Tr}(X) = 1 \\ & \mathbf{1}^T|X|\mathbf{1} \leq k \\ & X \succeq 0, \end{array}$$

to get a solution X_1, and truncate it to keep only the dominant (sparse) eigenvector x_1. Finally, we deflate A_1 to obtain

$$A_2 = A_1 - (x_1^T A_1 x_1) x_1 x_1^T,$$

and iterate to obtain further components.

The question is now: When do we stop the decomposition? In the PCA case, the decomposition stops naturally after $\mathbf{Rank}(A)$ factors have been found, since $A_{\mathbf{Rank}(A)+1}$ is then equal to zero. In the case of the sparse decomposition, we have no guarantee that this will happen. However, the robustness interpretation gives us a natural stopping criterion: if all the coefficients in $|A_i|$ are smaller than the noise level $\rho^\star$ (computed in the last section) then we must stop since the matrix is essentially indistinguishable from zero. So, even though we have no guarantee that the algorithm will terminate with a zero matrix, the decomposition will in practice terminate as soon as the coefficients in A become undistinguishable from the noise.

5 Algorithms

For problems of moderate size, our SDP can be solved efficiently using solvers such as SEDUMI [10] or SDPT3 [12]. For larger-scale problems, we need to resort to other types of algorithms for convex optimization. Of special interest are the recently-developed algorithms due to [9, 8, 1]. These are first-order methods specialized to problems having a specific saddle-point structure. It turns out that our problem, when expressed in the saddle-point form (6), falls precisely into this class of algorithms. Judged from the results presented in [8], in the closely related context of computing the Lovascz capacity of a graph, the theoretical complexity, as well as practical performance, of the method as applied to (6) should exhibit very significant improvements over the general-purpose interior-point algorithms for SDP. Of course, nothing comes without a price: for *fixed* problem size, the first-order methods mentioned above converge in $O(1/\epsilon)$, where ϵ is the required accuracy on the optimal value, while interior-point methods converge in $O(\log(1/\epsilon))$. In what follows, we adapt the algorithm in [8] to our particular constrained eigenvalue problem.

Given a $n \times n$ positive semi-definite symmetric matrix A, we consider the problem

$$\phi(A) := \max_U \ \mathbf{Tr}(AU) - \mathbf{1}^T|U|\mathbf{1} \ : \ U \succeq 0, \ \ \mathbf{Tr}\, U = 1. \tag{8}$$

By duality we have the representation

$$\begin{aligned} \phi(A) &= \min_{\|X\|_\infty \leq 1} \ \lambda_{\max}(A+X) \\ &= \min_{\|X\|_\infty \leq 1} \ \max_{U \in \Delta_n} \ \langle U, A+X \rangle \\ &= \min_{X \in \mathcal{Q}_1} \ f(X) \end{aligned}$$

where

$$\begin{gathered} \mathcal{Q}_1 = \{X \in \mathcal{S}^n \ : \ |X_{ij}| \leq 1, \ \ 1 \leq i,j \leq n\}, \\ f(X) = \lambda_{\max}(A+X) = \max_{U \in \mathcal{Q}_2} \langle AX, U \rangle - \hat{\phi}(U), \\ \mathcal{Q}_2 = \{U \in \mathcal{S}^n \ : \ \ \mathbf{Tr}\, U = 1\}, \ \ A = I_{n^2}, \ \ \hat{\phi}(U) = -\,\mathbf{Tr}(AU). \end{gathered}$$

Prox functions and related parameters. To $\mathcal{Q}_1$ and $\mathcal{Q}_2$ we associate norms and so-called prox-functions.

To $\mathcal{Q}_1$, we associate the Frobenius norm in in $\mathbf{R}^{n \times n}$, and a prox-function defined for $x \in \mathcal{Q}_1$ by

$$d_1(X) = \frac{1}{2} X^T X.$$

With this choice, the center X_0 of $\mathcal{Q}_1$, defined as

$$X_0 = \arg \min_{X \in \mathcal{Q}_1} \ d_1(X),$$

is $X_0 = 0$, and satisfies $d_1(X_0) = 0$. Moreover, we have

$$D_1 := \max_{X \in \mathcal{Q}_1} \ d_1(X) = n^2/2.$$

Furthermore, the function d_1 is strictly convex on its domain, with convexity parameter with respect to the Frobenius norm $\sigma_1 = 1$.

Next, for $\mathcal{Q}_2$ we use the dual of the standard matrix norm (denoted $\|\cdot\|_2^*$), and a prox-function

$$d_2(U) = \mathbf{Tr}(U \log U) + \log(n),$$

where $\log$ refers to the *matrix* (and not componentwise) logarithm. The center of the set $\mathcal{Q}_2$ is $X_0 = n^{-1} I_n$, and $d_2(X_0) = 0$. We have

$$\max_{U \in \mathcal{Q}_2} \ d_2(U) \leq \log n := D_2.$$

The convexity parameter of d_2 on its domain with respect to $\|\cdot\|_2^*$, is bounded below by $\sigma_2 = 1/2$. (This non-trivial result is proved in [8].)

Next we compute the $(1,2)$ norm of the operator A introduced above, which is defined as

$$\begin{aligned} \|A\|_{1,2} &:= \max_{X,U} \ \langle AX, U \rangle \ : \ \|X\|_F = 1, \ \|U\|_2^* = 1 \\ &= \max_X \ \|X\|_2 \ : \ \|X\|_F \leq 1 \\ &= 1. \end{aligned}$$

To summarize, the parameters set above are

$$D_1 = n^2/2, \ \ \sigma_1 = 1, \ \ D_2 = \log(n), \ \ \sigma_2 = 1/2, \ \ \|A\|_{1,2} = 1.$$

Idea of the method. The method first sets a regularization parameter

$$\mu = \frac{\epsilon}{2D_2}.$$

The method will produce an ϵ-suboptimal optimal value and corresponding sub-optimal solution in a number of steps not exceeding

$$N = 4\|A\|_{1,2}\sqrt{\frac{D_1 D_2}{\sigma_1 \sigma_2}} \cdot \frac{1}{\epsilon}.$$

The non-smooth objective of the original problem is replaced with

$$\min_{X \in \mathcal{Q}_1} \; f_\mu(X),$$

where f_μ is the penalized function involving the prox-function d_2:

$$f_\mu(X) = \max_{U \in \mathcal{Q}_2} \langle AX, U \rangle - \hat{\phi}(U) - \mu d_2(U).$$

Note that in our case, the function f_μ and its gradient are readily computed; we will detail this step later. The above function turns out to be a smooth uniform approximation to f everywhere, with maximal error $\mu D_2 = \epsilon/2$. Furthermore, the function f_μ is Lipschitz-continuous, with Lipschitz constant given by

$$L := \frac{D_2}{2\sigma_2} \cdot \frac{\|A\|_{1,2}^2}{\epsilon}.$$

In our specific case, the function f_μ is given by

$$f_\mu(X) = \mu \log \left(\mathbf{Tr} \exp((A + X)/\mu) \right) - \mu \log n,$$

which can be seen as a smooth approximation to the function $f(X) = \lambda_{\max}(A + X)$ that, for a specific choice of μ, enjoys nice uniform approximation properties with respect to f.

A specific gradient algorithm for smooth convex minimization is then applied to the above smooth convex function f_μ. The method requires the computation of values of

$$T_{\mathcal{Q}_1}(X) := \min_{Y \in \mathcal{Q}_1} \langle \nabla f_\mu(X), Y - X \rangle + \frac{1}{2} L \|X - Y\|_F^2,$$

where $X \in \mathcal{Q}_1$ is given. As seen later, in our case, the above problem essentially amounts to projecting on a box, and is easy.

The algorithm. Once the regularization parameter μ is set, the algorithm proceeds as follows.
For $k \geq 0$ **do**

1. Compute $f_\mu(X_k)$ and $\nabla f_\mu(X_k)$.
2. Find $Y_k = T_{\mathcal{Q}_1}(X_k)$.
3. Find $Z_k = \arg\min_X \left\{ \frac{L}{\sigma_1} d_1(X) + \sum_{i=0}^{k} \frac{i+1}{2} \langle \nabla f_\mu(X_i), X - X_i \rangle \; : \; X \in \mathcal{Q}_1 \right\}$.
4. Set $X_k = \frac{2}{k+3} Z_k + \frac{k+1}{k+3} Y_k$.

Note that the algorithm generates feasible points. Let us now detail the application of these steps to our specific problem. In what follows, the iteration count k is fixed and we denote X_k by X.

Step 1. The most expensive step in the algorithm is the first, namely computing function f_μ's values and gradient. For $Z = A + X$ a fixed $n \times n$ symmetric matrix, the problem boils down to computing

$$u^*(z) := \arg\max_{U \in \mathcal{Q}_2} \langle Z, U \rangle - \mu d_2(U) \tag{9}$$

associated optimal value $f_\mu(X)$. It turns out that this problem has a very simple solution, and only requires to form an eigenvalue decomposition for $Z = A + X$. The gradient of the objective function with respect to Z is set to the maximizer $u^*(Z)$ itself, so the gradient with respect to X is $\nabla f_\mu(X) = u^*(A + X)$.

To compute $u^*(Z)$, form an eigenvalue decomposition for Z: $z = VDV^T$, with $D = \mathbf{diag}(d)$. Then set, for $i = 1, \ldots, n$:

$$h_i = \frac{\exp(\frac{d_i - d_{\max}}{\mu})}{\sum_j \exp(\frac{d_j - d_{\max}}{\mu})}, \quad d_{\max} := \max_j d_j.$$

(In the above, $d_{\max}$ is used to prevent dealing with big numbers.) Then set $u^*(z) = VHV^T$, with $H = \mathbf{diag}(h)$. The corresponding function value is given by

$$f_\mu(X) = \mu \log(\mathbf{Tr} \exp((A + X)/\mu)) = \mu \log \left(\sum_{i=1}^n \exp(\frac{d_i}{\mu}) \right) - \mu \log n,$$

which can be reliably computed as before, as

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

Step 2. This step involves a problem of the form

$$T_{\mathcal{Q}_1}(X) = \arg\min_{Y \in \mathcal{Q}_1} \langle \nabla f_\mu(X), Y \rangle + \frac{1}{2} L \|X - Y\|_F^2,$$

where X is given. The above problem can be reduced to a projection:

$$\arg\min_{\|Y\|_\infty \leq 1} \|Y - V\|_F, \tag{10}$$

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

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

Step 3. The third step involves solving a problem of the same form as (10), with

$$V = -\frac{\sigma_1}{L} \cdot \sum_{i=0}^k \frac{i+1}{2} \nabla f_\mu(X_i).$$

Convergence criterion. We can stop the algorithm when the gap

$$\lambda_{\max}(A + X_k) - \mathbf{Tr}\, A U_k + \mathbf{1}^T |U_k| \mathbf{1} \leq \epsilon,$$

where $U_k = u^*((A + X_k)/\mu)$ is our current estimate of the dual variable (the function u^* is defined by (9)). The above gap is necessarily non-negative, since both X_k and U_k are feasible for the primal and dual problem, respectively. Nesterov advises to check this criterion only periodically, for example every 100 iterations.

Complexity. Since each iteration of the algorithm requires $O(n^3)$ flops, the predicted worst-case complexity to achieve an objective with absolute accuracy less than ϵ is

$$4\|A\|_{1,2}\sqrt{\frac{D_1 D_2}{\sigma_1\sigma_2}}\cdot\frac{O(n^3)}{\epsilon}. = O(n^4\sqrt{\log n}/\epsilon).$$

6 Numerical results

In this section, we illustrate the effectiveness of the proposed approach both on an artificial and a real-life data set. We compare with the other approaches mentioned in the introduction: PCA, PCA with simple thresholding, SCoTLASS and SPCA. The results show that our approach can achieve more sparsity in the principal components than SPCA does, while explaining as much variance. The other approaches can explain some more variance, but result in principal components that are far from sparse. We begin by a simple example illustrating the link between k and the cardinality of the solution.

6.1 Controlling sparsity with k

Here, we illustrate on a simple example how the sparsity of the solution to our relaxation evolves as k varies from 1 to n. We generate a 10×10 matrix U with uniformly distributed coefficients in $[0, 1]$. We let v be a sparse vector with:

$$v = (1, 0, 1, 0, 1, 0, 1, 0, 1, 0).$$

We then form a test matrix $A = U^TU + \sigma vv^T$, where σ is a signal-to-noise ratio equal to 15 in our case. We sample 50 different matrices A using this technique. For each k between 1 and 10 and each A, we solve the following SDP:

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

we then extract the first eigenvector of the solution X and record its cardinality. In Figure 1, we show the mean cardinality (and standard deviation) as a function of k. We observe that $k + 1$ is actually a good predictor of the cardinality, especially when $k + 1$ is close to the actual cardinality (5 in this case).

6.2 Artificial data

We consider the simulation example proposed by [15]. In this example, three hidden factors are created:

$$V_1 \sim \mathcal{N}(0, 290), \;\; V_2 \sim \mathcal{N}(0, 300), \;\; V_3 = -0.3V_1 + 0.925V_2 + \epsilon, \;\; \epsilon \sim \mathcal{N}(0, 300) \tag{11}$$

with V_1, V_2 and ϵ independent. Afterwards, 10 observed variables are generated as follows:

$$X_i = V_j + \epsilon_i^j, \;\; \epsilon_i^j \sim \mathcal{N}(0, 1),$$

with $j = 1$ for $i = 1, 2, 3, 4$, $j = 2$ for $i = 5, 6, 7, 8$ and $j = 3$ for $i = 9, 10$ and $\{\epsilon_i^j\}$ independent for $j = 1, 2, 3$, $i = 1, \ldots, 10$. Instead of sampling data from this model and computing an empirical covariance matrix of $(X_1, \ldots, X_{10})$, we use the exact covariance matrix to compute principal components using the different approaches.

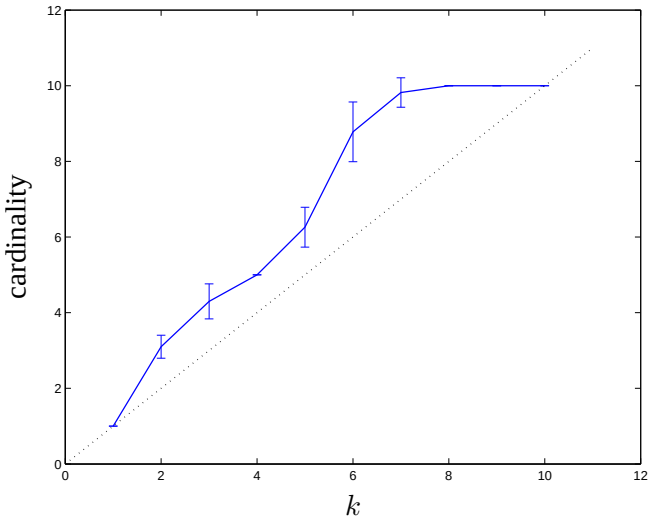


Figure 1: Cardinality versus k.

Since the three underlying factors have about the same variance, and the first two are associated with 4 variables while the last one is only associated with 2 variables, V_1 and V_2 are almost equally important, and they are both significantly more important than V_3. This, together with the fact that the first 2 principal components explain more than 99% of the total variance, suggests that considering two sparse linear combinations of the original variables should be sufficient to explain most of the variance in data sampled from this model. This is also discussed by [15]. The ideal solution would thus be to only use the variables (X_1, X_2, X_3, X_4) for the first sparse principal component, to recover the factor V_1, and only (X_5, X_6, X_7, X_8) for the second sparse principal component to recover V_2.

Using the true covariance matrix and the oracle knowledge that the ideal sparsity is 4, [15] performed SPCA (with $\lambda = 0$). We carry out our algorithm with $k = 4$. The results are reported in Table 1, together with results for PCA, simple thresholding and SCoTLASS ($t = 2$). Notice that SPCA, DSPCA and SCoTLASS all find the correct sparse principal components, while simple thresholding yields inferior performance. The latter wrongly includes the variables X_9 and X_{10} to explain most variance (probably it gets misled by the high correlation between V_2 and V_3), even more, it assigns higher loadings to X_9 and X_{10} than to one of the variables (X_5, X_6, X_7, X_8) that are clearly more important. Simple thresholding correctly identifies the second sparse principal component, probably because V_1 has a lower correlation with V_3. Simple thresholding also explains a bit less variance than the other methods.

6.3 Pit props data

The pit props data (consisting of 180 observations and 13 measured variables) was introduced by [4] and has become a standard example of the potential difficulty in interpreting principal components. [6] applied SCoTLASS to this problem and [15] used their SPCA approach, both with the goal of obtaining sparse principal components that can better be interpreted than those of PCA. SPCA performs better than SCoTLASS: it identifies principal components with respectively 7, 4, 4, 1, 1, and 1 non-zero loadings, as shown in Table 2. As shown in [15], this is much sparser than the modified principal components by SCoTCLASS, while explaining nearly the same variance (75.8% versus 78.2% for the 6 first principal components). Also, simple

thresholding of PCA, with a number of non-zero loadings that matches the result of SPCA, does worse than SPCA in terms of explained variance.

Following this previous work, we also consider the first 6 principal components. We try to identify principal components that are sparser than the best result of this previous work, i.e., SPCA, but explain the same variance. Therefore, we choose values for k of 5, 2, 2, 1, 1, 1 (two less than those of the SPCA results reported above, but no less than 1). Figure 2 shows the cumulative number of non-zero loadings and the cumulative explained variance (measuring the variance in the subspace spanned by the first i eigenvectors). The results for DSPCA are plotted with a red line and those for SPCA with a blue line. The cumulative explained variance for normal PCA is depicted with a black line. It can be seen that our approach is able to explain nearly the same variance as the SPCA method, while clearly reducing the number of non-zero loadings for the first 6 principal components. Adjusting the first k from 5 to 6 (relaxing the sparsity), we obtain the results plotted with a red dash-dot line: still better in sparsity, but with a cumulative explained variance that is fully competitive with SPCA. Moreover, as in the SPCA approach, the important variables associated with the 6 principal components do not overlap, which leads to a clearer interpretation. Table 2 shows the first three corresponding principal components for the different approaches (DSPCAw5 for $k_1 = 5$ and DSPCAw6 for $k_1 = 6$).

7 Conclusion

The semidefinite relaxation of the sparse principal component analysis problem proposed here appears to significantly improve the solution's sparsity, while explaining the same variance as previously proposed methods in the examples detailed above. The algorithms we used here handle moderate size problems efficiently. We are currently working on large-scale extensions using first-order techniques.

Acknowledgements

Thanks to Andrew Mullhaupt and Francis Bach for useful suggestions. We would like to acknowledge support from ONR MURI N00014-00-1-0637, Eurocontrol-C20052E/BM/03, NASA-NCC2-1428.

References

[1] A. Ben-Tal and A. Nemirovski. Non-euclidean restricted memory level method for large-scale convex optimization. *MINERVA Working paper*, 2004.

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

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

[4] J. Jeffers. Two case studies in the application of principal components. *Applied Statistics*, 16:225–236, 1967.

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

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

[7] Claude Lemaréchal and François Oustry. Semidefinite relaxations and Lagrangian duality with application to combinatorial optimization. *INRIA, Rapport de recherche*, 3710, 1999.

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	explained variance
PCA, PC1	.116	.116	.116	.116	-.395	-.395	-.395	-.395	-.401	-.401	60.0%
PCA, PC2	-.478	-.478	-.478	-.478	-.145	-.145	-.145	-.145	.010	.010	39.6%
ST, PC1	0	0	0	0	0	0	-.497	-.497	-.503	-.503	38.8%
ST, PC2	-.5	-.5	-.5	-.5	0	0	0	0	0	0	38.6%
other, PC1	0	0	0	0	.5	.5	.5	.5	0	0	40.9%
other, PC2	.5	.5	.5	.5	0	0	0	0	0	0	39.5%

Table 1: Loadings and explained variance for first two principal components, for the artificial example. 'ST' is the simple thresholding method, 'other' is all the other methods: SPCA, DSPCA and SCoTLASS.

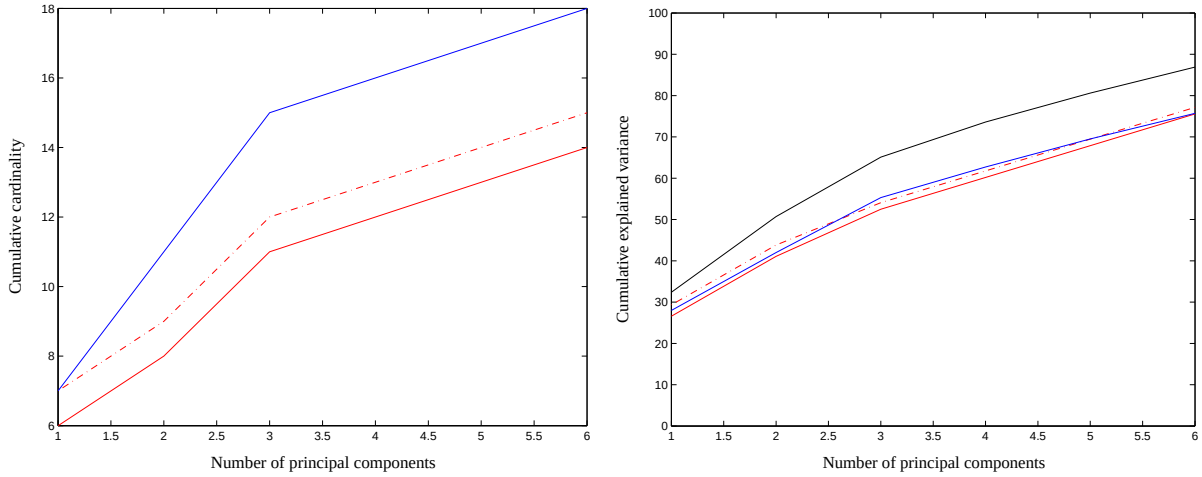


Figure 2: Cumulative cardinality and cumulative explained variance for SPCA and DSPCA as a function of the number of principal components: black line for normal PCA, blue for SPCA and red for DSPCA (full for $k_1 = 5$ and dash-dot for $k_1 = 6$).

	topdiam	length	moist	testsg	ovensg	ringtop	ringbud	bowmax	bowdist	whorls	clear	knots	diaknot
SPCA, PC1	-.477	-.476	0	0	.177	0	-.250	-.344	-.416	-.400	0	0	0
SPCA, PC2	0	0	.785	.620	0	0	0	-.021	0	0	0	.013	0
SPCA, PC3	0	0	0	0	.640	.589	.492	0	0	0	0	0	-.015
DSPCAw5, PC1	-.560	-.583	0	0	0	0	-.263	-.099	-.371	-.362	0	0	0
DSPCAw5, PC2	0	0	.707	.707	0	0	0	0	0	0	0	0	0
DSPCAw5, PC3	0	0	0	0	0	-.793	-.610	0	0	0	0	0	.012
DSPCAw6, PC1	-.491	-.507	0	0	0	-.067	-.357	-.234	-.387	-.409	0	0	0
DSPCAw6, PC2	0	0	.707	.707	0	0	0	0	0	0	0	0	0
DSPCAw6, PC3	0	0	0	0	0	-.873	-.484	0	0	0	0	0	.057

Table 2: Loadings for first three principal components, for the real-life example. DSPCAw5 (resp. DSPCAw6) shows the result of our technique with k_1 equal to 5 (resp. 6).

[8] A. Nemirovski. Prox-method with rate of convergence o(1/t) for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle-point problems. *MINERVA Working paper*, 2004.

[9] I. Nesterov. Smooth minimization of non-smooth functions. *CORE wroking paper*, 2003.

[10] Jos F. Sturm. Using sedumi 1.0x, a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11:625–653, 1999.

[11] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal statistical society, series B*, 58(267-288), 1996.

[12] K. C. Toh, M. J. Todd, and R. H. Tutuncu. Sdpt3 – a matlab software package for semidefinite programming. Technical report, School of Operations Research and Industrial Engineering, Cornell University, 1996.

[13] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38:49–95, 1996.

[14] S. Vines. Simple principal components. *Applied Statistics*, 49:441–451, 2000.

[15] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Technical report, statistics department, Stanford University*, 2004.