Journal of Machine Learning Research () Submitted 11/2008; Published

Generalized Power Method for Sparse Principal Component Analysis

Yurii Nesterov NESTEROV@CORE.UCL.AC.BE
Peter Richtárik RICHTARIK@CORE.UCL.AC.BE
Center for Operations Research and Econometrics
Catholic University of Louvain
Voie du Roman Pays 34
B-1348 Louvain-la-Neuve, Belgium

Michel Journée M.JOURNEE@ULG.AC.BE
Rodolphe Sepulchre R.SEPULCHRE@ULG.AC.BE
Dept. of Electrical Engineering and Computer Science
University of Liège
B-4000 Liège, Belgium

Editor:

Abstract

In this paper we develop a new approach to sparse principal component analysis (sparse PCA). We propose two single-unit and two block optimization formulations of the sparse PCA problem, aimed at extracting a single sparse dominant principal component of a data matrix, or more components at once, respectively. While the initial formulations involve nonconvex functions, and are therefore computationally intractable, we rewrite them into the form of an optimization program involving maximization of a convex function on a compact set. The dimension of the search space is decreased enormously if the data matrix has many more columns (variables) than rows. We then propose and analyze a simple gradient method suited for the task. It appears that our algorithm has best convergence properties in the case when either the objective function or the feasible set are strongly convex, which is the case with our single-unit formulations and can be enforced in the block case. Finally, we demonstrate numerically on a set of random and gene expression test problems that our approach outperforms existing algorithms both in quality of the obtained solution and in computational speed.

Keywords: sparse PCA, power method, gradient ascent, strongly convex sets, block algorithms

1. Introduction

Principal component analysis (PCA) is a well established tool for making sense of high dimensional data by reducing it to a smaller dimension. It has applications virtually in all areas of science—machine learning, image processing, engineering, genetics, neurocomputing, chemistry, meteorology, control theory, computer networks—to name just a few—where large data sets are encountered. It is important that having reduced dimension, the essential characteristics of the data are retained. If $A \in \mathbf{R}^{p\times n}$ is a matrix encoding p samples of n variables, with n being large, PCA aims at finding a few linear combinations of these variables, called *principal components*, which point in orthogonal directions explaining as much of the variance in the data as possible. If the variables contained in the columns of A are centered, then the classical PCA can be written in terms of the scaled *sample* *covariance matrix* $\Sigma = A^T A$ as follows:

$$\text{Find} \quad z^* = \arg \max_{z^T z \leq 1} z^T \Sigma z. \tag{1}$$

Extracting one component amounts to computing the dominant eigenvector of Σ (or, equivalently, dominant right singular vector of A). Full PCA involves the computation of the singular value decomposition (SVD) of A. Principal components are, in general, combinations of all the input variables, i.e. the *loading vector* z^* is not expected to have many zero coefficients. In most applications, however, the original variables have concrete physical meaning and PCA then appears especially interpretable if the extracted components are composed only from a small number of the original variables. In the case of gene expression data, for instance, each variable represents the expression level of a particular gene. A good analysis tool for biological interpretation should be capable to highlight “simple” structures in the genome—structures expected to involve a few genes only—that explain a significant amount of the specific biological processes encoded in the data. Components that are linear combinations of a small number of variables are, quite naturally, usually easier to interpret. It is clear, however, that with this additional goal, some of the explained variance has to be sacrificed. The objective of *sparse principal component analysis* (sparse PCA) is to find a reasonable *trade-off* between these conflicting goals. One would like to explain *as much* variability in the data as possible, using components constructed from *as few* variables as possible. This is the classical trade-off between *statistical fidelity* and *interpretability*.

For about a decade, sparse PCA has been a topic of active research. Historically, the first suggested approaches were based on ad-hoc methods involving post-processing of the components obtained from classical PCA. For example, Jolliffe (1995) consider using various rotation techniques to find sparse loading vectors in the subspace identified by PCA. Cadima and Jolliffe (1995) proposed to simply set to zero the PCA loadings which are in absolute value smaller than some threshold constant.

In recent years, more involved approaches have been put forward – approaches that consider the conflicting goals of explaining variability and achieving representation sparsity simultaneously. These methods usually cast the sparse PCA problem in the form of an optimization program, aiming at maximizing explained variance penalized for the number of non-zero loadings. For instance, the SCoTLASS algorithm proposed by Jolliffe et al. (2003) aims at maximizing the Rayleigh quotient of the covariance matrix of the data under the ℓ_1-norm based Lasso penalty (Tibshirani (1996)). Zou et al. (2006) formulate sparse PCA as a regression-type optimization problem and impose the Lasso penalty on the regression coefficients. d’Aspremont et al. (2007) in their DSPCA algorithm exploit convex optimization tools to solve a convex relaxation of the sparse PCA problem. Shen and Huang (2008) adapt the singular value decomposition (SVD) to compute low-rank matrix approximations of the data matrix under various sparsity-inducing penalties. Greedy methods, which are typical for combinatorial problems, have been investigated by Moghaddam et al. (2006). Finally, d’Aspremont et al. (2008) proposed a greedy heuristic accompanied with a certificate of optimality.

In many applications, several components need to be identified. The traditional approach consists of incorporating an existing single-unit algorithm in a deflation scheme, and computing the desired number of components sequentially (see, e.g., d’Aspremont et al. (2007)). In the case of Rayleigh quotient maximization it is well-known that computing several components at once instead of computing them one-by-one by deflation with the classical power method might present better convergence whenever the largest eigenvalues of the underlying matrix are close to each other (see, e.g., Parlett (1980)). Therefore, block approaches for sparse PCA are expected to be more efficient on ill-posed problems.

In this paper we consider two single-unit (Section 2.1 and 2.2) and two block formulations (Section 2.3 and 2.4) of sparse PCA, aimed at extracting m sparse principal components, with $m = 1$ in the former case and $p \geq m > 1$ in the latter. Each of these two groups comes in two variants, depending on the type of penalty we use to enforce sparsity—either ℓ_1 or ℓ_0 (cardinality).[1] Although we assume a direct access to the data matrix A, these formulations also hold when only the covariance matrix Σ is available, provided that a factorization of the form $\Sigma = A^T A$ is identified (e.g., by eigenvalue decomposition or by Cholesky decomposition).

While our basic formulations involve maximization of a *nonconvex* function on a space of dimension involving n, we construct *reformulations* that cast the problem into the form of maximization of a *convex* function on the unit Euclidean sphere in $\mathbf{R}^p$ (in the $m = 1$ case) or the *Stiefel manifold*[2] in $\mathbf{R}^{p\times m}$ (in the $m > 1$ case). The advantage of the reformulation becomes apparent when trying to solve problems with many variables ($n \gg p$), since we manage to avoid searching a space of large dimension.[3] At the same time, due to the convexity of the new cost function we are able to propose and *analyze* the iteration-complexity of a simple gradient-type scheme, which appears to be well suited for problems of this form. In particular, we study (Section 3) a first-order method for solving an optimization problem of the form

$$f^* = \max_{x\in\mathcal{Q}} f(x), \qquad \text{(P)}$$

where $\mathcal{Q}$ is a compact subset of a finite-dimensional vector space and f is convex. It appears that our method has best theoretical convergence properties when either f or $\mathcal{Q}$ are strongly convex, which is the case in the single unit case (unit ball is strongly convex) and can be enforced in the block case by adding a strongly convex regularizing term to the objective function, constant on the feasible set. We do not, however, prove any results concerning the quality of the obtained solution. Even the goal of obtaining a local maximizer is in general unattainable, and we must be content with convergence to a stationary point.

In the particular case when $\mathcal{Q}$ is the unit Euclidean ball in $\mathbf{R}^p$ and $f(x) = x^T Cx$ for some $p \times p$ symmetric positive definite matrix C, our gradient scheme specializes to the *power method*, which aims at maximizing the *Rayleigh quotient*

$$R(x) = \frac{x^T Cx}{x^T x}$$

and thus at computing the largest eigenvalue, and the corresponding eigenvector, of C.

By applying our general gradient scheme to our sparse PCA reformulations of the form (P), we obtain algorithms (Section 4) with per-iteration computational cost $\mathcal{O}(npm)$.

1. Our single-unit cardinality-penalized formulation is identical to that of d'Aspremont et al. (2008).
2. Stiefel manifold is the set of rectangular matrices with orthonormal columns.
3. Note that in the case $p > n$, it is recommended to factor the covariance matrix as $\Sigma = A^T A$ with $A \in \mathbf{R}^{n\times n}$, such that the dimension p in the reformulations equals at most n.

We demonstrate on random Gaussian (Section 5.1) and gene expression data related to breast cancer (Section 5.2) that our methods are very efficient in practice. While achieving a balance between the explained variance and sparsity which is the same as or superior to the existing methods, they are faster, often converging before some of the other algorithms manage to initialize. Additionally, in the case of gene expression data our approach seems to extract components with strongest biological content.

Notation. For convenience of the reader, and at the expense of redundancy, some of the less standard notation below is also introduced at the appropriate place in the text where it is used. Parameters $m \leq p \leq n$ are actual values of dimensions of spaces used in the paper. In the definitions below, we use these actual values (i.e. n, p and m) if the corresponding object we define is used in the text exclusively with them; otherwise we make use of the dummy variables k (representing p or n in the text) and l (representing m, p or n in the text).

Given a vector $y \in \mathbf{R}^k$, its i^{th} coordinate is denoted by y_i. With a slight abuse of notation, the same notation y_i is used for the i^{th} column of a matrix $Y \in \mathbf{R}^{k\times l}$, and y_{ij} is the element of Y at position (i,j).

By $\mathbf{E}$ we refer to a finite-dimensional vector space; $\mathbf{E}^*$ is its conjugate space, i.e. the space of all linear functionals on $\mathbf{E}$. By $\langle s, x\rangle$ we denote the action of $s \in \mathbf{E}^*$ on $x \in \mathbf{E}$. For a self-adjoint positive definite linear operator $G : \mathbf{E} \to \mathbf{E}^*$ we define a pair of norms on $\mathbf{E}$ and $\mathbf{E}^*$ as follows

$$\begin{aligned} \|x\| &\overset{\text{def}}{=} \langle Gx, x\rangle^{1/2}, \quad x \in \mathbf{E}, \\ \|s\|_* &\overset{\text{def}}{=} \langle s, G^{-1}s\rangle^{1/2}, \quad s \in \mathbf{E}^*. \end{aligned} \tag{2}$$

Although the theory in Section 3 is developed in this general setting, the sparse PCA applications considered in this paper require either the choice $\mathbf{E} = \mathbf{E}^* = \mathbf{R}^p$ (see Section 3.3 and problems (8) and (14) in Section 2) or $\mathbf{E} = \mathbf{E}^* = \mathbf{R}^{p\times m}$ (see Section 3.4 and problems (18) and (22) in Section 2). In both cases we will let G be the corresponding identity operator for which we obtain

$$\langle x, y\rangle = \sum_i x_i y_i, \quad \|x\| = \langle x, x\rangle^{1/2} = \left(\sum_i x_i^2\right)^{1/2} = \|x\|_2, \quad x, y \in \mathbf{R}^p, \text{and}$$

$$\langle X, Y\rangle = \operatorname{Tr} X^T Y, \quad \|X\| = \langle X, X\rangle^{1/2} = \left(\sum_{ij} x_{ij}^2\right)^{1/2} = \|X\|_F, \quad X, Y \in \mathbf{R}^{p\times m}.$$

Thus in the vector setting we work with the *standard Euclidean norm* and in the matrix setting with the *Frobenius norm*. The symbol Tr denotes the trace of its argument.

Furthermore, for $z \in \mathbf{R}^n$ we write $\|z\|_1 = \sum_i |z_i|$ (ℓ_1 norm) and by $\|z\|_0$ (ℓ_0 "norm") we refer to the number of nonzero coefficients, or *cardinality*, of z. By $\mathbf{S}^p$ we refer to the space of all $p \times p$ symmetric matrices; $\mathbf{S}^p_+$ (resp. $\mathbf{S}^p_{++}$) refers to the positive semidefinite (resp. definite) cone. Eigenvalues of matrix Y are denoted by $\lambda_i(Y)$, largest eigenvalue by $\lambda_{\max}(Y)$. Analogous notation with the symbol σ refers to singular values.

By $\mathcal{B}^k = \{y \in \mathbf{R}^k \mid y^T y \leq 1\}$ (resp. $\mathcal{S}^k = \{y \in \mathbf{R}^k \mid y^T y = 1\}$) we refer to the unit Euclidean ball (resp. sphere) in $\mathbf{R}^k$. If we write $\mathcal{B}$ and $\mathcal{S}$, then these are the corresponding objects in $\mathbf{E}$. The space of $n \times m$ matrices with unit-norm columns will be denoted by

$$[\mathcal{S}^n]^m = \{Y \in \mathbf{R}^{n\times m} \mid \operatorname{Diag}(Y^T Y) = I_m\},$$

where $\mathrm{Diag}(\cdot)$ represents the diagonal matrix obtained by extracting the diagonal of the argument. *Stiefel manifold* is the set of rectangular matrices of fixed size with orthonormal columns:

$$\mathcal{S}_m^p = \{Y \in \mathbf{R}^{p\times m} \mid Y^TY = I_m\}.$$

For $t \in \mathbf{R}$ we will further write $\mathrm{sign}(t)$ for the sign of the argument and $t_+ = \max\{0, t\}$.

2. Some formulations of the sparse PCA problem

In this section we propose four formulations of the sparse PCA problem, all in the form of the general optimization framework (P). The first two deal with the single-unit sparse PCA problem and the remaining two are their generalizations to the block case.

2.1 Single-unit sparse PCA via ℓ_1-penalty

Let us consider the optimization problem

$$\phi_{\ell_1}(\gamma) \overset{\text{def}}{=} \max_{z\in\mathcal{B}^n} \sqrt{z^T\Sigma z} - \gamma\|z\|_1, \tag{3}$$

with sparsity-controlling parameter $\gamma \geq 0$ and sample covariance matrix $\Sigma = A^TA$.

The solution $z^*(\gamma)$ of (3) in the case $\gamma = 0$ is equal to the right singular vector corresponding to $\sigma_{\max}(A)$, the largest singular value of A. It is the first principal component of the data matrix A. The optimal value of the problem is thus equal to

$$\phi_{\ell_1}(0) = (\lambda_{\max}(A^TA))^{1/2} = \sigma_{\max}(A).$$

Note that there is no reason to expect this vector to be sparse. On the other hand, for large enough γ, we will necessarily have $z^*(\gamma) = 0$, obtaining maximal sparsity. Indeed, since

$$\max_{z\neq 0} \frac{\|Az\|_2}{\|z\|_1} = \max_{z\neq 0} \frac{\|\sum_i z_i a_i\|_2}{\|z\|_1} \leq \max_{z\neq 0} \frac{\sum_i |z_i|\|a_i\|_2}{\sum_i |z_i|} = \max_i \|a_i\|_2 = \|a_{i^*}\|_2,$$

we get $\|Az\|_2 - \gamma\|z\|_1 < 0$ for all nonzero vectors z whenever γ is chosen to be strictly bigger than $\|a_{i^*}\|_2$. From now on we will assume that

$$\gamma < \|a_{i^*}\|_2. \tag{4}$$

Note that there is a trade-off between the value $\|Az^*(\gamma)\|_2$ and the sparsity of the solution $z^*(\gamma)$. The penalty parameter γ is introduced to "continuously" interpolate between the two extreme cases described above, with values in the interval $[0, \|a_{i^*}\|_2)$. It depends on the particular application whether sparsity is valued more than the explained variance, or vice versa, and to what extent. Due to these considerations, we will consider the solution of (3) to be a sparse principal component of A.

Reformulation. The reader will observe that the objective function in (3) is not convex, nor concave, and that the feasible set is of a high dimension if $p \ll n$. It turns out that these shortcomings are overcome by considering the following reformulation:

$$\begin{aligned}\phi_{\ell_1}(\gamma) &= \max_{z\in\mathcal{B}^n} \|Az\|_2 - \gamma\|z\|_1 \\ &= \max_{z\in\mathcal{B}^n}\max_{x\in\mathcal{B}^p} x^TAz - \gamma\|z\|_1 \qquad (5)\\ &= \max_{x\in\mathcal{B}^p}\max_{z\in\mathcal{B}^n} \sum_{i=1}^n z_i(a_i^Tx) - \gamma|z_i| \\ &= \max_{x\in\mathcal{B}^p}\max_{\bar{z}\in\mathcal{B}^n} \sum_{i=1}^n |\bar{z}_i|(|a_i^Tx| - \gamma), \qquad (6)\end{aligned}$$

where $z_i = \operatorname{sign}(a_i^Tx)\bar{z}_i$. In view of (4), there is some $x \in \mathcal{B}^n$ for which $a_i^Tx > \gamma$. Fixing such x, solving the inner maximization problem for $\bar{z}$ and then translating back to z, we obtain the closed-form solution

$$z_i^* = z_i^*(\gamma) = \frac{\operatorname{sign}(a_i^Tx)[|a_i^Tx| - \gamma]_+}{\sqrt{\sum_{k=1}^n [|a_k^Tx| - \gamma]_+^2}}, \quad i = 1,\ldots,n. \qquad (7)$$

Problem (6) can therefore be written in the form

$$\boxed{\phi_{\ell_1}^2(\gamma) = \max_{x\in\mathcal{S}^p} \sum_{i=1}^n [|a_i^Tx| - \gamma]_+^2.} \qquad (8)$$

Note that the objective function is differentiable and convex, and hence all local and global maxima must lie on the boundary, i.e., on the unit Euclidean sphere $\mathcal{S}^p$. Also, in the case when $p \ll n$, formulation (8) requires to search a space of a much lower dimension than the initial problem (3).

Sparsity. In view of (7), an optimal solution x^* of (8) defines a sparsity pattern of the vector z^*. In fact, the coefficients of z^* indexed by

$$\mathcal{I} = \{i \mid |a_i^Tx^*| > \gamma\} \qquad (9)$$

are active while all others must be zero. Geometrically, active indices correspond to the defining hyperplanes of the polytope

$$\mathcal{D} = \{x \in \mathbf{R}^p \mid |a_i^Tx| \leq 1\}$$

that are (strictly) crossed by the line joining the origin and the point x^*/γ. Note that it is possible to say something about the sparsity of the solution even without the knowledge of x^*:

$$\gamma \geq \|a_i\|_2 \quad \Rightarrow \quad z_i^*(\gamma) = 0, \qquad i = 1,\ldots,n. \qquad (10)$$

2.2 Single-unit sparse PCA via cardinality penalty

Instead of the ℓ_1-penalization, d'Aspremont et al. (2008) consider the formulation

$$\phi_{\ell_0}(\gamma) \overset{\text{def}}{=} \max_{z\in\mathcal{B}^n} z^T\Sigma z - \gamma\,\|z\|_0, \qquad (11)$$

which directly penalizes the number of nonzero components (cardinality) of the vector z.

Reformulation. The reasoning of the previous section suggests the reformulation

$$\phi_{\ell_0}(\gamma) = \max_{x\in\mathcal{B}^p} \max_{z\in\mathcal{B}^n} (x^T A z)^2 - \gamma\|z\|_0, \tag{12}$$

where the maximization with respect to $z \in \mathcal{B}^n$ for a fixed $x \in \mathcal{B}^p$ has the closed form solution

$$z_i^* = z_i^*(\gamma) = \frac{[\operatorname{sign}((a_i^T x)^2 - \gamma)]_+ a_i^T x}{\sqrt{\sum_{k=1}^n [\operatorname{sign}((a_k^T x)^2 - \gamma)]_+ (a_k^T x)^2}}, \quad i = 1, \dots, n. \tag{13}$$

In analogy with the ℓ_1 case, this derivation assumes that

$$\gamma < \|a_{i^*}\|_2^2,$$

so that there is $x \in \mathcal{B}^n$ such that $(a_i^T x)^2 - \gamma > 0$. Otherwise $z^* = 0$ is optimal. Formula (13) is easily obtained by analyzing (12) separately for fixed cardinality values of z. Hence, problem (11) can be cast in the following form

$$\boxed{\phi_{\ell_0}(\gamma) = \max_{x\in\mathcal{S}^p} \sum_{i=1}^n [(a_i^T x)^2 - \gamma]_+.} \tag{14}$$

Again, the objective function is convex, albeit nonsmooth, and the new search space is of particular interest if $p \ll n$. A different derivation of (14) for the $n = p$ case can be found in d'Aspremont et al. (2008).

Sparsity. Given a solution x^* of (14), the set of active indices of z^* is given by

$$\mathcal{I} = \{i \mid (a_i^T x^*)^2 > \gamma\}.$$

Geometrically, active indices correspond to the defining hyperplanes of the polytope

$$\mathcal{D} = \{x \in \mathbf{R}^p \mid |a_i^T x| \leq 1\}$$

that are (strictly) crossed by the line joining the origin and the point $x^*/\sqrt{\gamma}$. As in the ℓ_1 case, we have

$$\gamma \geq \|a_i\|_2^2 \quad \Rightarrow \quad z_i^*(\gamma) = 0, \qquad i = 1, \dots, n. \tag{15}$$

2.3 Block sparse PCA via ℓ_1-penalty

Consider the following block generalization of (5),

$$\phi_{\ell_1,m}(\gamma) \overset{\text{def}}{=} \max_{\substack{X\in\mathcal{S}_m^p \\ Z\in[\mathcal{S}^n]^m}} \operatorname{Tr}(X^T A Z N) - \sum_{j=1}^m \gamma_j \sum_{i=1}^n |z_{ij}|, \tag{16}$$

where the m-dimensional vector $\gamma = [\gamma_1, \dots \gamma_m]^T$ is nonnegative and $N = \operatorname{Diag}(\mu_1, \dots, \mu_m)$, with positive entries on the diagonal. The dimension m corresponds to the number of extracted components and is assumed to be smaller or equal to the rank of the data matrix, i.e., $m \leq \mathrm{Rank}(A)$. Each parameter γ_j controls the sparsity of the corresponding component. It will be shown below that under some conditions on the parameters μ_j, the case $\gamma = 0$ recovers PCA. In that particular instance, any solution Z^* of (16) has orthonormal columns, although this is not explicitly enforced. For positive γ_j, the columns of Z^* are not expected to be orthogonal anymore. Most existing algorithms for computing several sparse principal components, e.g., Zou et al. (2006); d'Aspremont et al. (2007); Shen and Huang (2008), also do not impose orthogonal loading directions. Simultaneously enforcing sparsity and orthogonality seems to be a hard (and perhaps questionable) task.

Reformulation. Since problem (16) is completely decoupled in the columns of Z, i.e.,

$$\phi_{\ell_1,m}(\gamma) = \max_{X \in \mathcal{S}_m^p} \sum_{j=1}^{m} \max_{z_j \in \mathcal{S}^n} \mu_j x_j^T A z_j - \gamma_j \|z_j\|_1,$$

the closed-form solution (7) of (5) is easily adapted to the block formulation (16):

$$z_{ij}^* = z_{ij}^*(\gamma_j) = \frac{\mathrm{sign}(a_i^T x_j)[\mu_j |a_i^T x_j| - \gamma_j]_+}{\sqrt{\sum_{k=1}^n [\mu_j |a_k^T x_j| - \gamma_j]_+^2}}. \tag{17}$$

This leads to the reformulation

$$\boxed{\phi_{\ell_1,m}^2(\gamma) = \max_{X \in \mathcal{S}_m^p} \sum_{j=1}^{m} \sum_{i=1}^{n} [\mu_j |a_i^T x_j| - \gamma_j]_+^2,} \tag{18}$$

which maximizes a convex function $f : \mathbf{R}^{p \times m} \to \mathbf{R}$ on the Stiefel manifold $\mathcal{S}_m^p$.

Sparsity. A solution X^* of (18) again defines the sparsity pattern of the matrix Z^*: the entry z_{ij}^* is active if

$$\mu_j |a_i^T x_j^*| > \gamma_j,$$

and equal to zero otherwise. For all $\gamma_j > \mu_j \max_i \|a_i\|_2$, the trivial solution $Z^* = 0$ is optimal.

Block PCA. For $\gamma = 0$, problem (18) can be equivalently written in the form

$$\phi_{\ell_1,m}^2(0) = \max_{X \in \mathcal{S}_m^p} \mathrm{Tr}(X^T A A^T X N^2), \tag{19}$$

which has been well studied (see e.g., Brockett (1991) and Absil et al. (2008)). The solutions of (19) span the dominant m-dimensional invariant subspace of the matrix AA^T. Furthermore, if the parameters μ_j are all distinct, the columns of X^* are the m dominant eigenvectors of AA^T, i.e., the m dominant left-eigenvectors of the data matrix A. The columns of the solution Z^* of (16) are thus the m dominant right singular vectors of A, i.e., the PCA loading vectors. Such a matrix N with distinct diagonal elements enforces the objective function in (19) to have isolated maximizers. In fact, if $N = I_m$, any point $X^* U$ with X^* a solution of (19) and $U \in \mathcal{S}_m^m$ is also a solution of (19). In the case of sparse PCA, i.e., $\gamma > 0$, the penalty term already ensures isolated maximizers, such that the diagonal elements of N do not have to be distinct. However, as it will be briefly illustrated in the forthcoming numerical experiments (Section 5), having distinct elements on the diagonal of N pushes towards sparse loading vectors that are more orthogonal.

2.4 Block sparse PCA via cardinality penalty

The single-unit cardinality-penalized case can also be naturally extended to the block case:

$$\phi_{\ell_0,m}(\gamma) \overset{\text{def}}{=} \max_{\substack{X\in\mathcal{S}_m^p \\ Z\in[\mathcal{S}^n]^m}} \operatorname{Tr}(\operatorname{Diag}(X^TAZN)^2) - \sum_{j=1}^m \gamma_j \|z_j\|_0, \tag{20}$$

where the sparsity inducing vector $\gamma = [\gamma_1, \dots \gamma_m]^T$ is nonnegative and $N = \operatorname{Diag}(\mu_1, \dots, \mu_m)$ with positive entries on the diagonal. In the case $\gamma = 0$, problem (22) is equivalent to (19) and therefore corresponds to PCA, provided that all μ_j are distinct.

Reformulation. Again, this block formulation is completely decoupled in the columns of Z,

$$\phi_{\ell_0,m}(\gamma) = \max_{X\in\mathcal{S}_m^p} \sum_{j=1}^m \max_{z_j\in\mathcal{S}^n} (\mu_j x_j^T A z_j)^2 - \gamma_j \|z_j\|_0,$$

so that the solution (13) of the single unit case provides the optimal columns z_i:

$$z_{ij}^* = z_{ij}^*(\gamma_j) = \frac{[\operatorname{sign}((\mu_j a_i^T x_j)^2 - \gamma_j)]_+ \mu_j a_i^T x_j}{\sqrt{\sum_{k=1}^n [\operatorname{sign}((\mu_j a_k^T x_j)^2 - \gamma_j)]_+ \mu_j^2 (a_k^T x_j)^2}}. \tag{21}$$

The reformulation of problem (20) is thus

$$\boxed{\phi_{\ell_0,m}(\gamma) = \max_{X\in\mathcal{S}_m^p} \sum_{j=1}^m \sum_{i=1}^n [(\mu_j a_i^T x_j)^2 - \gamma_j]_+,} \tag{22}$$

which maximizes a convex function $f : \mathbf{R}^{p\times m} \to \mathbf{R}$ on the Stiefel manifold $\mathcal{S}_m^p$.

Sparsity. For a solution X^* of (22), the active entries z_{ij}^* of Z^* are given by the condition

$$(\mu_j a_i^T x_j^*)^2 > \gamma_j.$$

Hence for all $\gamma_j > \mu_j^2 \max_i \|a_i\|_2^2$, the optimal solution of (20) is $Z^* = 0$.

3. A gradient method for maximizing convex functions

By $\mathbf{E}$ we denote an arbitrary finite-dimensional vector space; $\mathbf{E}^*$ is its conjugate, i.e. the space of all linear functionals on $\mathbf{E}$. We equip these spaces with norms given by (2).

In this section we propose and analyze a simple gradient-type method for maximizing a convex function $f : \mathbf{E} \to \mathbf{R}$ on a compact set $\mathcal{Q}$:

$$\boxed{f^* = \max_{x\in\mathcal{Q}} f(x).} \tag{23}$$

Unless explicitly stated otherwise, we will *not* assume f to be differentiable. By $f'(x)$ we denote any subgradient of function f at x. By $\partial f(x)$ we denote its subdifferential.

At any point $x \in \mathcal{Q}$ we introduce some measure for the first-order optimality conditions:

$$\Delta(x) \stackrel{\text{def}}{=} \max_{y \in \mathcal{Q}} \langle f'(x), y - x \rangle.$$

It is clear that

$$\Delta(x) \geq 0, \tag{24}$$

with equality only at those points x where the gradient $f'(x)$ belongs to the normal cone to the set $\mathrm{Conv}(\mathcal{Q})$ at x.[4]

3.1 Algorithm

Consider the following simple algorithmic scheme.

Algorithm 1: Gradient scheme

```
input  : $x_0 \in \mathcal{Q}$
output: $x_k$ (approximate solution of (23))
begin
    $k \longleftarrow 0$
    repeat
        $x_{k+1} \in \mathrm{Arg\,max}\{f(x_k) + \langle f'(x_k), y - x_k \rangle \mid y \in \mathcal{Q}\}$
        $k \longleftarrow k + 1$
    until a stopping criterion is satisfied
end
```

Note that for example in the special case $\mathcal{Q} = r\mathcal{S} \stackrel{\text{def}}{=} \{x \in \mathbf{E} \mid \|x\| = r\}$ or $\mathcal{Q} = r\mathcal{B} \stackrel{\text{def}}{=} \{x \in \mathbf{E} \mid \|x\| \leq r\}$, the main step of Algorithm 1 can be written in an explicit form:

$$x_{k+1} = r\frac{G^{-1}f'(x_k)}{\|f'(x_k)\|_*}. \tag{25}$$

3.2 Analysis

Our first convergence result is straightforward. Denote $\Delta_k \stackrel{\text{def}}{=} \min_{0 \leq i \leq k} \Delta(x_i)$.

Theorem 1 *Let sequence $\{x_k\}_{k=0}^{\infty}$ be generated by Algorithm 1 as applied to a convex function f. Then the sequence $\{f(x_k)\}_{k=0}^{\infty}$ is monotonically increasing and $\lim_{k\to\infty} \Delta(x_k) = 0$. Moreover,*

$$\Delta_k \leq \frac{f^* - f(x_0)}{k+1}. \tag{26}$$

Proof From convexity of f we immediately get

$$f(x_{k+1}) \geq f(x_k) + \langle f'(x_k), x_{k+1} - x_k \rangle = f(x_k) + \Delta(x_k), \tag{27}$$

4. The normal cone to the set $\mathrm{Conv}(\mathcal{Q})$ at $x \in \mathcal{Q}$ is *smaller* than the normal cone to the set $\mathcal{Q}$. Therefore, the optimality condition $\Delta(x) = 0$ is *stronger* than the standard one.

and therefore, $f(x_{k+1}) \geq f(x_k)$ for all k. By summing up these inequalities for $k = 0, 1, \ldots, N-1$, we obtain

$$f^* - f(x_0) \geq f(x_k) - f(x_0) \geq \sum_{i=0}^{k} \Delta(x_i),$$

and the result follows. ■

For a sharper analysis, we need some technical assumptions on f and $\mathcal{Q}$.

Assumption 1 *The norms of the subgradients of f are bounded from below on $\mathcal{Q}$ by a positive constant, i.e.*

$$\delta_f \overset{def}{=} \min_{\substack{x \in \mathcal{Q} \\ f'(x) \in \partial f(x)}} \|f'(x)\|_* > 0. \tag{28}$$

This assumption is not too binding because of the following result.

Proposition 2 *Assume that there exists a point $\bar{x} \notin \mathcal{Q}$ such that $f(\bar{x}) < f(x)$ for all $x \in \mathcal{Q}$. Then*

$$\delta_f \geq \left[\min_{x \in \mathcal{Q}} f(x) - f(\bar{x})\right] \Big/ \left[\max_{x \in \mathcal{Q}} \|x - \bar{x}\|\right] > 0.$$

Proof Because f is convex, for any $x \in \mathcal{Q}$ we have

$$0 < f(x) - f(\bar{x}) \leq \langle f'(x), x - \bar{x} \rangle \leq \|f'(x)\|_* \|x - \bar{x}\|.$$

■

For our next convergence result we need to assume either strong convexity of f or strong convexity of the set $\mathrm{Conv}(\mathcal{Q})$.

Assumption 2 *Function f is* strongly convex, *i.e. there exists a constant $\sigma_f > 0$ such that for any $x, y \in \mathbf{E}$*

$$f(y) \geq f(x) + \langle f'(x), y - x \rangle + \frac{\sigma_f}{2} \|y - x\|^2. \tag{29}$$

Note that convex functions satisfy inequality (29) with *convexity parameter* $\sigma_f = 0$.

Assumption 3 *The set $\mathrm{Conv}(\mathcal{Q})$ is* strongly convex, *i.e. there is a constant $\sigma_{\mathcal{Q}} > 0$ such that for any $x, y \in \mathrm{Conv}(\mathcal{Q})$ and $\alpha \in [0, 1]$ the following inclusion holds:*

$$\alpha x + (1 - \alpha) y + \frac{\sigma_{\mathcal{Q}}}{2} \alpha (1 - \alpha) \|x - y\|^2 \mathcal{S} \subset \mathrm{Conv}(\mathcal{Q}). \tag{30}$$

Note that any set $\mathcal{Q}$ satisfies inclusion (30) with *convexity parameter* $\sigma_{\mathcal{Q}} = 0$.

It can be shown (see the Appendix), that level sets of strongly convex functions with Lipschitz continuous gradient are again strongly convex. An example of such a function is the simple quadratic $x \mapsto \|x\|^2$. The level sets of this function correspond to Euclidean balls of varying sizes.

As we will see in Theorem 4, a better analysis of Algorithm 1 is possible if $\mathrm{Conv}(\mathcal{Q})$, the convex hull of the feasible set of problem (23), is strongly convex. Note that in the case of the two formulations (8) and (14) of the sparse PCA problem, the feasible set $\mathcal{Q}$ is the unit Euclidean sphere. Since the convex hull of the unit sphere is the unit ball, which is a strongly convex set, the feasible set of our sparse PCA formulations satisfies Assumption 3.

In the special case $\mathcal{Q} = r\mathcal{S}$ for some $r > 0$, there is a simple proof that Assumption 3 holds with $\sigma_{\mathcal{Q}} = \frac{1}{r}$. Indeed, for any $x, y \in \mathbf{E}$ and $\alpha \in [0,1]$, we have

$$\begin{aligned} \|\alpha x + (1-\alpha)y\|^2 &= \alpha^2\|x\|^2 + (1-\alpha)^2\|y\|^2 + 2\alpha(1-\alpha)\langle Gx, y\rangle \\ &= \alpha\|x\|^2 + (1-\alpha)\|y\|^2 - \alpha(1-\alpha)\|x-y\|^2. \end{aligned}$$

Thus, for $x, y \in r\mathcal{S}$ we obtain

$$\|\alpha x + (1-\alpha)y\| = \left[r^2 - \alpha(1-\alpha)\|x-y\|^2\right]^{1/2} \le r - \frac{1}{2r}\alpha(1-\alpha)\|x-y\|^2.$$

Hence, we can take $\sigma_{\mathcal{Q}} = \frac{1}{r}$.

The relevance of Assumption 3 is justified by the following technical observation.

Proposition 3 *If f be convex, then for any two subsequent iterates x_k, x_{k+1} of Algorithm 1*

$$\Delta(x_k) \ge \frac{\sigma_{\mathcal{Q}}}{2}\|f'(x_k)\|_*\|x_{k+1} - x_k\|^2. \tag{31}$$

Proof We have noted in (24) that for convex f we have $\Delta(x_k) \ge 0$. We can thus concentrate on the situation when $\sigma_{\mathcal{Q}} > 0$ and $f'(x_k) \neq 0$. Note that

$$\langle f'(x_k), x_{k+1} - y\rangle \ge 0 \quad \text{for all} \quad y \in \operatorname{Conv}(\mathcal{Q}).$$

We will use this inequality with

$$y = y_\alpha \overset{\text{def}}{=} x_k + \alpha(x_{k+1} - x_k) + \frac{\sigma_{\mathcal{Q}}}{2}\alpha(1-\alpha)\|x_{k+1} - x_k\|^2 \frac{G^{-1}f'(x_k)}{\|f'(x_k)\|_*}, \ \alpha \in [0,1].$$

In view of (30), $y_\alpha \in \operatorname{Conv}(\mathcal{Q})$, and therefore

$$0 \ge \langle f'(x_k), y_\alpha - x_{k+1}\rangle = (1-\alpha)\langle f'(x_k), x_k - x_{k+1}\rangle + \frac{\sigma_{\mathcal{Q}}}{2}\alpha(1-\alpha)\|x_{k+1} - x_k\|^2\|f'(x_k)\|_*.$$

Since α is an arbitrary value from $[0,1]$, the result follows. ■

We are now ready to refine our analysis of Algorithm 1.

Theorem 4 (Stepsize convergence) *Let f be convex ($\sigma_f \ge 0$), and let either Assumption 2 ($\sigma_f > 0$) or Assumptions 1 ($\delta_f > 0$) and 3 ($\delta_{\mathcal{Q}} > 0$) be satisfied. If $\{x_k\}$ is the sequence of points generated by Algorithm 1, then*

$$\sum_{k=0}^{\infty} \|x_{k+1} - x_k\|^2 \le \frac{2(f^* - f(x_0))}{\sigma_{\mathcal{Q}}\delta_f + \sigma_f}. \tag{32}$$

Proof Since f is convex, Proposition 3 gives

$$f(x_{k+1}) - f(x_k) \geq \Delta(x_k) + \frac{\sigma_f}{2}\|x_{k+1} - x_k\|^2 \geq \frac{1}{2}(\sigma_Q \delta_f + \sigma_f)\|x_{k+1} - x_k\|^2.$$

The additional assumptions of the theorem ensure that $\sigma_Q \delta_f + \delta_f > 0$. It remains to add the inequalities up for $k \geq 0$. ■

Theorem 4 gives an upper estimate on the number of iterations it takes for Algorithm 1 to produce a step of small size. Indeed,

$$k \geq \frac{2(f^* - f(x_0))}{\sigma_Q \delta_f + \sigma_f}\frac{1}{\epsilon^2} - 1 \quad \Rightarrow \quad \min_{0 \leq i \leq k} \|x_{i+1} - x_i\| \leq \epsilon.$$

It can be illustrated on simple examples that it is not in general possible to guarantee that the algorithm will produce iterates converging to a local maximizer. However, Theorem 4 guarantees that the set of the limit points is connected, and that all of them satisfy the first-order optimality condition. Also notice that, started from a local minimizer, the method will not move away.

Termination. A reasonable stopping criterion for Algorithm 1 is the following: terminate once the relative change of the objective function becomes small:

$$\frac{f(x_{k+1}) - f(x_k)}{f(x_k)} \leq \epsilon, \qquad \text{or equivalently,} \qquad f(x_{k+1}) \leq (1+\epsilon) f(x_k). \tag{33}$$

3.3 Maximization with spherical constraints

Consider $\mathbf{E} = \mathbf{E}^* = \mathbf{R}^p$ with $G = I_p$ and $\langle s, x \rangle = \sum_i s^{(i)} x^{(i)}$, and let

$$\mathcal{Q} = r\mathcal{S}^p = \{x \in \mathbf{R}^p \mid \|x\| = r\}.$$

Problem (23) takes on the form

$$\boxed{f^* = \max_{x \in r\mathcal{S}^p} f(x).} \tag{34}$$

Since $\mathcal{Q}$ is strongly convex ($\sigma_{\mathcal{Q}} = \frac{1}{r}$), Theorem 4 is meaningful for any convex function f ($\sigma_f \geq 0$). The main step of Algorithm 1 can be written down explicitly (see (25)):

$$x_{k+1} = r\frac{f'(x_k)}{\|f'(x_k)\|_2}.$$

The following examples illustrate the connection of Algorithm 1 to classical methods.

Example 5 (Power method) *In the special case of a quadratic objective function $f(x) = \frac{1}{2}x^T Cx$ for some $C \in \mathbf{S}^p_{++}$ on the unit sphere ($r = 1$), we have*

$$f^* = \tfrac{1}{2}\lambda_{max}(C),$$

and Algorithm 1 is equivalent to the power iteration method *for computing the largest eigenvalue of C (Golub and Van Loan (1996)). Hence for $\mathcal{Q} = \mathcal{S}^p$, we can think of our scheme as a generalization of the power method. Indeed, our algorithm performs the following iteration:*

$$x_{k+1} = \frac{Cx_k}{\|Cx_k\|}, \quad k \geq 0.$$

Note that both δ_f and σ_f are equal to the smallest eigenvalue of C, and hence the right-hand side of (32) is equal to

$$\frac{\lambda_{max}(C) - x_0^T C x_0}{2\lambda_{min}(C)}. \tag{35}$$

Example 6 (Shifted power method) *If C is not positive semidefinite in the previous example, the objective function is not convex and our results are not applicable. However, this complication can be circumvented by instead running the algorithm with the shifted quadratic function*

$$\hat{f}(x) = \frac{1}{2} x^T (C + \omega I_p) x,$$

where $\omega > 0$ satisfies $\hat{C} = \omega I_p + C \in \mathbf{S}_{++}^p$. On the feasible set, this change only adds a constant term to the objective function. The method, however, produces different sequence of iterates. Note that the constants δ_f and σ_f are also affected and, correspondingly, the estimate (35).

The example above illustrates an easy "trick" to make strongly convex an objective function that it is not: one simply adds to the original objective function a strongly convex function that is constant on the boundary of the feasible set. The two formulations are equivalent since the objective functions differ only by a constant on the domain of interest. However, there is a clear trade-off. If the second term dominates the first term (say, by choosing very large ω), the algorithm will tend to treat the objective as a quadratic, and will hence tend to terminate in fewer iterations, nearer to the starting iterate. In the limit case, the method will not move away from the initial iterate. This issue deserves futher analysis.

3.4 Maximization with orthonormality constraints

Consider $\mathbf{E} = \mathbf{E}^* = \mathbf{R}^{p \times m}$, the space of $p \times m$ real matrices, with $m \leq p$. Note that for $m = 1$ we recover the setting of the previous section. We assume this space is equipped with the trace inner product: $\langle X, Y \rangle = \operatorname{Tr}(X^T Y)$. The induced norm, denoted by $\|X\|_F \overset{\text{def}}{=} \langle X, X \rangle^{1/2}$, is the Frobenius norm (we let G be the identity operator). We can now consider various feasible sets, the simplest being a ball or a sphere. Due to nature of applications in this paper, let us concentrate on the situation when $\mathcal{Q}$ is a special subset of the sphere with radius $r = \sqrt{m}$, the Stiefel manifold:

$$\mathcal{Q} = \mathcal{S}_m^p = \{X \in \mathbf{R}^{p \times m} \mid X^T X = I_m\}.$$

Problem (23) then takes on the following form:

$$\boxed{f^* = \max_{X \in \mathcal{S}_m^p} f(X).}$$

Using the duality of the nuclear and spectral matrix norms and Proposition 7 below it can be shown that $\operatorname{Conv}(\mathcal{Q})$ is equal to the unit spectral ball. It can be then further deduced that this set is not strongly convex ($\sigma_{\mathcal{Q}} = 0$) and as a consequence, Theorem 4 is meaningful only if f is strongly convex ($\sigma_f > 0$). Of course, Theorem 1 applies also in the $\sigma_f = 0$ case.

At every iteration, Algorithm 1 needs to maximize a linear function over the Stiefel manifold. In the text that follows, it will be convenient to use the symbol $\operatorname{Polar}(C)$ for the U factor of the *polar decomposition* of matrix $C \in \mathbf{R}^{p \times m}$:

$$C = UP, \qquad U \in \mathcal{S}_m^p, \qquad P \in \mathbf{S}_+^m.$$

The complexity of the polar decomposition is $\mathcal{O}(pm^2)$, with $p \geq m$. In view of the Proposition 7, the main step of Algorithm 1 can be written in the form

$$x_{k+1} = \mathrm{Polar}(f'(x_k)). \tag{36}$$

Proposition 7 *Let $C \in \mathbf{R}^{p\times m}$, with $m \leq p$, and denote by $\sigma_i(C)$, $i = 1, \ldots, m$, the singular values of C. Then*

$$\max_{X\in\mathcal{S}^p_m} \langle C, X\rangle = \sum_{i=1}^m \sigma_i(C) \qquad (= \|C\|_* = \mathrm{Tr}[(C^TC)^{1/2}]), \tag{37}$$

with maximizer $X^* = \mathrm{Polar}(C)$. If C is of full rank, then $\mathrm{Polar}(C) = C(C^TC)^{-1/2}$.

Proof Existence of the polar factorization in the nonsquare case is covered by Theorem 7.3.2 in Horn and Johnson (1985). Let $C = V\Sigma W^T$ be the singular value decomposition (SVD) of A; that is, V is $p \times p$ orthonormal, W is $m \times m$ orthonormal, and Σ is $p \times m$ diagonal with values $\sigma_i(A)$ on the diagonal. Then

$$\begin{aligned}
\max_{X\in\mathcal{S}^p_m} \langle C, X\rangle &= \max_{X\in\mathcal{S}^p_m} \langle V\Sigma W^T, X\rangle \\
&= \max_{X\in\mathcal{S}^p_m} \langle \Sigma, V^TXW\rangle \\
&= \max_{Z\in\mathcal{S}^p_m} \langle \Sigma, Z\rangle = \max_{Z\in\mathcal{S}^p_m} \sum_{i=1}^m \sigma_i(C) z^{(ii)} \leq \sum_i^m \sigma_i(C).
\end{aligned}$$

The third equality follows since the function $X \mapsto V^TXW$ maps $\mathcal{S}^p_m$ onto itself. Both factors of the polar decomposition of C can be easily read-off from the SVD. Indeed, if we let $\bar{V}$ be the submatrix of V consisting of its first m columns and $\bar{\Sigma}$ be the principal $m \times m$ submatrix of Σ, i.e. a diagonal matrix with values $\sigma_i(C)$ on its diagonal, then $C = \bar{V}\bar{\Sigma}W^T = (\bar{V}W^T)(W\bar{\Sigma}W^T)$ and we can put $U = \bar{V}W^T$ and $P = W\bar{\Sigma}W^T$. To establish (37) it remains to note that

$$\langle C, U\rangle = \mathrm{Tr}\, P = \sum_i \lambda_i(P) = \sum_i \sigma_i(P) = \mathrm{Tr}(P^TP)^{1/2} = \mathrm{Tr}(C^TC)^{1/2} = \sum_i \sigma_i(C).$$

Finally, since $C^TC = PU^TUP = P^2$, we have $P = (C^TC)^{1/2}$, and in the full rank case we obtain $X^* = U = CP^{-1} = C(C^TC)^{-1/2}$. ■

Note that the block sparse PCA formulations (18) and (22) conform to this setting. Here is one more example:

Example 8 (Rectangular Procrustes Problem) *Let $C, X \in \mathbf{R}^{p\times m}$ and $D \in \mathbf{R}^{p\times p}$ and consider the following problem:*

$$\min\{\|C - DX\|_F^2 \mid X^TX = I_m\}. \tag{38}$$

Since $\|C - DX\|_F^2 = \|C\|_F^2 + \langle DX, DX\rangle - 2\langle CD, X\rangle$, by a similar shifting technique as in the previous example we can cast problem (38) in the following form

$$\max\{\omega\|X\|_F^2 - \langle DX, DX\rangle + 2\langle CD, X\rangle \mid X^TX = I_m\}.$$

For $\omega > 0$ large enough, the new objective function will be strongly convex. In this case our algorithm becomes similar to the gradient method proposed in Fraikin et al. (2008).

The standard Procrustes problem in the literature is a special case of (38) with $p = m$.

4. Algorithms for sparse PCA

The solutions of the sparse PCA formulations of Section 2 provide locally optimal patterns of zeros and nonzeros for a vector $z \in \mathcal{S}^n$ (in the single-unit case) or a matrix $Z \in [\mathcal{S}^n]^m$ (in the block case). The sparsity-inducing penalty term used in these formulations biases however the values assigned to the nonzero entries, which should be readjusted by considering the sole objective of maximum variance. An algorithm for sparse PCA combines thus a method that identifies a "good" pattern of sparsity with a method that fills the active entries. In the sequel, we discuss the general block sparse PCA problem. The single-unit case is recovered in the particular case $m = 1$.

4.1 Methods for pattern-finding

The application of our general method (Algorithm 1) to the four sparse PCA formulations of Section 2, i.e., (8), (14), (18) and (22), leads to Algorithms 2, 3, 4 and 5 below, that provide a locally optimal pattern of sparsity for a matrix $Z \in [\mathcal{S}^n]^m$. This pattern is defined as a binary matrix $P \in \{0,1\}^{n \times m}$ such that $p_{ij} = 1$ if the loading z_{ij} is active and $p_{ij} = 0$ otherwise. So P is an indicator of the coefficients of Z that are zeroed by our method. The computational complexity of the single-unit algorithms (Algorithms 2 and 3) is $\mathcal{O}(np)$ operations per iteration. The block algorithms (Algorithms 4 and 5) have complexity $\mathcal{O}(npm)$ per iteration.

These algorithms need to be initialized at a point for which the associated sparsity pattern has *at least one* active element. In case of the single-unit algorithms, such an initial iterate $x \in \mathcal{S}^p$ is chosen parallel to the column of A with the largest norm, i.e.,

$$x = \frac{a_{i^*}}{\|a_{i^*}\|_2}, \quad \text{where} \quad i^* = \arg\max_i \|a_i\|_2. \tag{39}$$

For the block algorithms, a suitable initial iterate $X \in \mathcal{S}^p_m$ is constructed in a block-wise manner as $X = [x|X_\perp]$, where x is the unit-norm vector (39) and $X_\perp \in \mathcal{S}^p_{m-1}$ is orthogonal to x, i.e., $x^T X_\perp = 0$.

The nonnegative parameters γ have to be chosen below the upper bounds derived in Section 2 and which are summarized in Table 1. Increasing the value of these parameters leads to solutions of smaller cardinality. There is however not explicit relationship between γ and the resulting cardinality. Since the proposed algorithms are fast, one can afford some trials and errors to reach a targeted cardinality. We however see it as an advantage not to enforce a fixed cardinality, since this information is often unknown a priori. As illustrated in the forthcoming numerical experiments (Section 5), our algorithms are able to recover cardinalities that are best adapted to the model that underlies the data.

As previously explained, the parameters μ_j required by the block algorithms can be either identical (e.g., equal to one) or distinct (e.g., $\mu_j = \frac{1}{j}$). Since distinct μ_j leads to orthogonal loading vectors in the PCA case (i.e., $\gamma = 0$), they are expected to push towards orthogonality also in the sparse PCA case. Nevertheless, unless otherwise stated, the technical parameters μ_j will be set to one in what follows.

Algorithm 2	Single-unit ℓ_1	$\gamma \leq \max_i \|a_i\|_2$
Algorithm 3	Single-unit ℓ_0	$\gamma \leq \max_i \|a_i\|_2^2$
Algorithm 4	Block ℓ_1	$\gamma_j \leq \mu_j \max_i \|a_i\|_2$
Algorithm 5	Block ℓ_0	$\gamma_j \leq \mu_j^2 \max_i \|a_i\|_2^2$

Table 1: Theoretical upper-bounds on the sparsity parameters γ.

Let us finally mention that the input matrix A of these algorithms can be the data matrix itself as well as any matrix such that the factorization $\Sigma = A^T A$ of the covariance matrix holds. This property is very valuable when there is no access to the data and only the covariance matrix is available, or when the number of samples is greater than the number of variables. In this last case, the dimension p can be reduced to at most n by computing an eigenvalue decomposition or a Cholesky decomposition of the covariance matrix, for instance.

Algorithm 2: Single-unit sparse PCA method based on the ℓ_1-penalty (8)

```
input  : Data matrix $A \in \mathbf{R}^{p \times n}$
         Sparsity-controlling parameter $\gamma \geq 0$
         Initial iterate $x \in \mathcal{S}^p$
output: A locally optimal sparsity pattern $P$
begin
    repeat
        $x \longleftarrow \sum_{i=1}^n [|a_i^T x| - \gamma]_+ \operatorname{sign}(a_i^T x) a_i$
        $x \longleftarrow \frac{x}{\|x\|}$
    until a stopping criterion is satisfied
    Construct vector $P \in \{0,1\}^n$ such that $\begin{cases} p_i = 1 & \text{if } |a_i^T x| > \gamma \\ p_i = 0 & \text{otherwise.} \end{cases}$
end
```

Algorithm 3: Single-unit sparse PCA algorithm based on the ℓ_0-penalty (14)

```
input  : Data matrix $A \in \mathbf{R}^{p \times n}$
         Sparsity-controlling parameter $\gamma \geq 0$
         Initial iterate $x \in \mathcal{S}^p$
output: A locally optimal sparsity pattern $P$
begin
    repeat
        $x \longleftarrow \sum_{i=1}^n [\operatorname{sign}((a_i^T x)^2 - \gamma)]_+ \, a_i^T x \, a_i$
        $x \longleftarrow \frac{x}{\|x\|}$
    until a stopping criterion is satisfied
    Construct vector $P \in \{0,1\}^n$ such that $\begin{cases} p_i = 1 & \text{if } (a_i^T x)^2 > \gamma \\ p_i = 0 & \text{otherwise.} \end{cases}$
end
```

Algorithm 4: Block Sparse PCA algorithm based on the ℓ_1-penalty (18)

```
input  : Data matrix $A \in \mathbf{R}^{p\times n}$
         Sparsity-controlling vector $[\gamma_1, \dots \gamma_m]^T \geq 0$
         Parameters $\mu_1, \dots, \mu_m > 0$
         Initial iterate $X \in \mathcal{S}^p_m$
output: A locally optimal sparsity pattern $P$
begin
    repeat
        for $j = 1, \dots, m$ do
            $x_j \longleftarrow \sum_{i=1}^n \mu_j [\mu_j |a_i^T x_j| - \gamma_j]_+ \operatorname{sign}(a_i^T x) a_i$
        $X \longleftarrow \operatorname{Polar}(X)$
    until a stopping criterion is satisfied
    Construct matrix $P \in \{0,1\}^{n\times m}$ such that $\begin{cases} p_{ij} = 1 & \text{if } \mu_j |a_i^T x_j| > \gamma_j \\ p_{ij} = 0 & \text{otherwise.} \end{cases}$
end
```

Algorithm 5: Block Sparse PCA algorithm based on the ℓ_0-penalty (22)

```
input  : Data matrix $A \in \mathbf{R}^{p\times n}$
         Sparsity-controlling vector $[\gamma_1, \dots \gamma_m]^T \geq 0$
         Parameters $\mu_1, \dots, \mu_m > 0$
         Initial iterate $X \in \mathcal{S}^p_m$
output: A locally optimal sparsity pattern $P$
begin
    repeat
        for $j = 1, \dots, m$ do
            $x_j \longleftarrow \sum_{i=1}^n \mu_j^2 [\operatorname{sign}((\mu_j a_i^T x_j)^2 - \gamma_j)]_+ \, a_i^T x_j \, a_i$
        $X \longleftarrow \operatorname{Polar}(X)$
    until a stopping criterion is satisfied
    Construct matrix $P \in \{0,1\}^{n\times m}$ such that $\begin{cases} p_{ij} = 1 & \text{if } (\mu_j a_i^T x_j)^2 > \gamma_j \\ p_{ij} = 0 & \text{otherwise.} \end{cases}$
end
```

4.2 Post-processing

Once a "good" sparsity pattern P has been identified, the active entries of Z still have to be filled. To this end, we consider the optimization problem,

$$(X^*, Z^*) \overset{\text{def}}{=} \arg \max_{\substack{X \in \mathcal{S}^p_m \\ Z \in [\mathcal{S}^n]^m \\ Z_{\bar{P}} = 0}} \operatorname{Tr}(X^T A Z N), \tag{40}$$

where $\bar{P} \in \{0,1\}^{n\times m}$ is the complement of P, $Z_{\bar{P}}$ denotes the entries of Z that are constrained to zero and $N = \operatorname{Diag}(\mu_1, \dots, \mu_m)$ with strictly positive μ_i. Problem (40) assigns the active part of the loading vectors Z to maximize the variance explained by the resulting components. Without loss of generality, each column of P is assumed to contain active elements.

In the single-unit case $m = 1$, an explicit solution of (40) is available,

$$\begin{aligned} X^* &= u, \\ Z_P^* &= v \text{ and } Z_{\bar{P}}^* = 0, \end{aligned} \tag{41}$$

where $\sigma u v^T$ with $\sigma > 0$, $u \in \mathcal{B}^p$ and $v \in \mathcal{B}^{\|P\|_0}$ is a rank one singular value decomposition of the matrix A_P, that corresponds to the submatrix of A containing the columns related to the active entries. The post-processing (41) is equivalent to the *variational renormalization* proposed by Moghaddam et al. (2006).

Although an exact solution of (40) is hard to compute in the block case $m > 1$, a local maximizer can be efficiently computed by optimizing alternatively with respect to one variable while keeping the other ones fixed. The following lemmas provide an explicit solution to each of these subproblems.

Lemma 9 *For a fixed $Z \in [\mathcal{S}^n]^m$, a solution X^* of*

$$\max_{X \in \mathcal{S}_m^p} \operatorname{Tr}(X^T A Z N)$$

is provided by the U factor of the polar decomposition of the product AZN.

Proof See Proposition 7. ■

Lemma 10 *The solution*

$$Z^* \overset{def}{=} \arg \max_{\substack{Z \in [\mathcal{S}^n]^m \\ Z_{\bar{P}} = 0}} \operatorname{Tr}(X^T A Z N), \tag{42}$$

is at any point $X \in \mathcal{S}_m^p$ defined by the two conditions $Z_P^* = (A^T X N D)_P$ and $Z_{\bar{P}}^* = 0$, where D is a positive diagonal matrix that normalizes each column of Z^* to unit norm, i.e.,

$$D = \operatorname{Diag}(N X^T A A^T X N)^{-\frac{1}{2}}.$$

Proof The Lagrangian of the optimization problem (42) is

$$\mathcal{L}(Z, \Lambda_1, \Lambda_2) = \operatorname{Tr}(X^T A Z N) - \operatorname{Tr}(\Lambda_1 (Z^T Z - I_m)) - \operatorname{Tr}(\Lambda_2^T Z),$$

where the Lagrangian multipliers $\Lambda_1 \in \mathbf{R}^{m \times m}$ and $\Lambda_2 \in \mathbf{R}^{n \times m}$ have the following properties: Λ_1 is an invertible diagonal matrix and $(\Lambda_2)_P = 0$. The first order optimality conditions of (42) are thus

$$\begin{aligned} A^T X N - 2 Z \Lambda_1 - \Lambda_2 &= 0 \\ \operatorname{Diag}(Z^T Z) &= I_m \\ Z_P &= 0. \end{aligned}$$

Hence, any stationary point Z^* of (42) satisfies $Z^*_P = (A^T XND)_P$ and $Z^*_{\bar{P}} = 0$, where D is a diagonal matrix that normalizes the columns of Z^* to unit norm. The second order optimality condition imposes the diagonal matrix D to be positive. Such a D is unique and given by $D = \mathrm{Diag}(NX^T AA^T XN)^{-\frac{1}{2}}$. ■

The alternating optimization scheme is summarized in Algorithm 6, which computes a local solution of (40). A judicious initialization is provided by an accumulation point of the algorithm for pattern-finding, i.e., Algorithms 4 and 5.

Algorithm 6: Alternating optimization scheme for solving (40)

```
input  : Data matrix $A \in \mathbf{R}^{p\times n}$
         Sparsity pattern $P \in \{0,1\}^{n\times m}$
         Matrix $N = \mathrm{Diag}(\mu_1, \ldots, \mu_m)$
         Initial iterate $X \in \mathcal{S}^p_m$
output: A local minimizer $(X, Z)$ of (40)
begin
    repeat
        $Z \longleftarrow A^T XN$
        $Z \longleftarrow Z\ \mathrm{Diag}(Z^T Z)^{-\frac{1}{2}}$
        $Z_{\bar{P}} \longleftarrow 0$
        $X \longleftarrow \mathrm{Polar}(AZN)$
    until a stopping criterion is satisfied
end
```

It should be noted that Algorithm 6 is a postprocessing heuristic that, strictly speaking, is required only for the ℓ_1 block formulation (Algorithm 4). In fact, since the cardinality penalty only depends on the sparsity pattern P and not on the actual values assigned to Z_P, a solution (X^*, Z^*) of Algorithms 3 or 5 is also a local maximizer of (40) for the resulting pattern P. This explicit solution provides a good alternative to Algorithm 6. In the single unit case with ℓ_1 penalty (Algorithm 2), the solution (41) is available.

4.3 Sparse PCA algorithms

To sum up, in this paper we propose four sparse PCA algorithms, each combining a method to identify a "good" sparsity pattern with a method to fill the active entries of the m loading vectors. They are summarized in Table 2. The MATLAB code of these GPower[5] algorithms is available on the authors' websites.[6]

5. Our algorithms are named GPower where the "G" stands for *generalized* or *gradient*.
6. http://www.inma.ucl.ac.be/∼richtarik
http://www.montefiore.ulg.ac.be/∼journee

	Computation of P	Computation of Z_P
GPower_{ℓ_1}	Algorithm 2	Equation (41)
GPower_{ℓ_0}	Algorithm 3	Equation (13)
$\mathsf{GPower}_{\ell_1,m}$	Algorithm 4	Algorithm 6
$\mathsf{GPower}_{\ell_0,m}$	Algorithm 5	Equation (21)

Table 2: New algorithms for sparse PCA.

4.4 Deflation scheme.

For the sake of completeness, we recall a classical deflation process for computing m sparse principal components with a single-unit algorithm (d'Aspremont et al. (2007)). Let $z \in \mathbf{R}^n$ be a unit-norm sparse loading vector of the data A. Subsequent directions can be sequentially obtained by computing a dominant sparse component of the residual matrix $A - xz^T$, where $x = Az$ is the vector that solves

$$\min_{x\in\mathbf{R}^p} \|A - xz^T\|_F.$$

Further deflation techniques for sparse PCA have been proposed by Mackey (2008).

4.5 Connection with existing sparse PCA methods

As previously mentioned, our ℓ_0-based single-unit algorithm GPower_{ℓ_0} rests on the same reformulation (14) as the greedy algorithm proposed by d'Aspremont et al. (2008).

There is also a clear connection between both single-unit algorithms GPower_{ℓ_1} and GPower_{ℓ_0} and the rSVD algorithms of Shen and Huang (2008), which solve the optimization problems

$$\min_{\substack{z\in\mathbf{R}^n \\ x\in\mathcal{S}^p}} \|A - xz^T\|_F^2 + 2\gamma\|z\|_1 \quad \text{and} \quad \min_{\substack{z\in\mathbf{R}^n \\ x\in\mathcal{S}^p}} \|A - xz^T\|_F^2 + \gamma\|z\|_0$$

by alternating optimization over one variable (either x or z) while fixing the other one. It can be shown that an update on x amounts to the iterations of Algorithms 2 and 3, depending on the penalty type. Although rSVD and GPower were derived differently, it turns out that, except for the initialization and post-processing phases, the algorithms *are identical*. There are, however, several benefits to our approach: 1) we are able to analyze convergence properties of the method, 2) we show that the core algorithm can be derived as a special case of a generalization of the power method (and hence more applications are possible), 3) we give generalizations from single unit case to block case, 4) our approach uncovers the possibility of a very useful initialization technique, 5) we equip the method with a practical postprocessing phase, 6) we provide a link with the formulation of d'Aspremont et al. (2008).

5. Numerical experiments

In this section, we evaluate the proposed power algorithms against existing sparse PCA methods. Three competing methods are considered in this study: a greedy scheme aimed at computing a local maximizer of (11) (Approximate Greedy Search Algorithm, d'Aspremont et al. (2008)), the SPCA algorithm (Zou et al. (2006)) and the sPCA-rSVD algorithm (Shen and Huang (2008)). We do not include the DSPCA algorithm (d'Aspremont et al. (2007)) in our numerical study. This method solves a convex relaxation of the sparse PCA problem and has a large per-iteration computational complexity of $\mathcal{O}(n^3)$ compared to the other methods. Table 3 lists the considered algorithms.

GPower_{ℓ_1}	Single-unit sparse PCA via ℓ_1-penalty
GPower_{ℓ_0}	Single-unit sparse PCA via ℓ_0-penalty
$\mathsf{GPower}_{\ell_1,m}$	Block sparse PCA via ℓ_1-penalty
$\mathsf{GPower}_{\ell_0,m}$	Block sparse PCA via ℓ_0-penalty
Greedy	Greedy method
SPCA	SPCA algorithm
rSVD_{ℓ_1}	sPCA-rSVD algorithm with an ℓ_1-penalty ("soft thresholding")
rSVD_{ℓ_0}	sPCA-rSVD algorithm with an ℓ_0-penalty ("hard thresholding")

Table 3: Sparse PCA algorithms we compare in this section.

These algorithms are compared on random data (Sections 5.1 and 5.2) as well as on real data (Sections 5.3 and 5.4). All numerical experiments are performed in MATLAB. The parameter ϵ in the stopping criterion of the GPower algorithms has been fixed to 10^{-4}. MATLAB implementations of the SPCA algorithm and the greedy algorithm have been rendered available by Zou et al. (2006) and d'Aspremont et al. (2008). We have, however, implemented the sPCA-rSVD algorithm on our own (Algorithm 1 in Shen and Huang (2008)), and use it with the same stopping criterion as for the GPower algorithms. This algorithm initializes with the best rank-one approximation of the data matrix. This is done by a first run of the algorithm with the sparsity-inducing parameter γ that is set to zero.

Given a data matrix $A \in \mathbf{R}^{p\times n}$, the considered sparse PCA algorithms provide m unit-norm sparse loading vectors stored in the matrix $Z \in [\mathcal{S}^n]^m$. The samples of the associated components are provided by the m columns of the product AZ. The variance explained by these m components is an important comparison criterion of the algorithms. In the simple case $m = 1$, the variance explained by the component Az is

$$\mathrm{Var}(z) = z^T A^T A z.$$

When z corresponds to the first principal loading vector, the variance is $\mathrm{Var}(z) = \sigma_{\max}(A)^2$. In the case $m > 1$, the derived components are likely to be correlated. Hence, summing up the variance explained individually by each of the components overestimates the variance explained simultaneously by all the components. This motivates the notion of *adjusted variance* proposed by Zou et al. (2006). The adjusted variance of the m components $Y = AZ$ is defined as

$$\mathrm{AdjVar}\ Z = \mathrm{Tr}\, R^2,$$

where $Y = QR$ is the QR decomposition of the components sample matrix Y ($Q \in \mathcal{S}^p_m$ and R is an $m \times m$ upper triangular matrix).

5.1 Random data drawn from a sparse PCA model

In this section, we follow the procedure proposed by Shen and Huang (2008) to generate random data with a covariance matrix having sparse eigenvectors. To this end, a covariance matrix is first synthesized through the eigenvalue decomposition $\Sigma = VDV^T$, where the first m columns of $V \in \mathbf{R}^{n\times n}$ are pre-specified sparse orthonormal vectors. A data matrix $A \in \mathbf{R}^{p\times n}$ is then generated by drawing p samples from a zero-mean normal distribution with covariance matrix Σ, i.e., $A \sim N(0, \Sigma)$.

Consider a setup with $n = 500, m = 2$ and $p = 50$, where the two orthonormal eigenvectors are specified as follows

$$\begin{cases} v_{1i} = \frac{1}{\sqrt{10}} & \text{for} \quad i = 1, \ldots, 10, \\ v_{1i} = 0 & \text{otherwise,} \end{cases} \qquad \begin{cases} v_{2i} = \frac{1}{\sqrt{10}} & \text{for} \quad i = 11, \ldots, 20, \\ v_{2i} = 0 & \text{otherwise,} \end{cases}$$

The remaining eigenvectors $v_j,\ j > 2$, are chosen arbitrarily, and the eigenvalues are fixed at the following values

$$\begin{cases} d_{11} = 400 \\ d_{22} = 300 \\ d_{jj} = 1, \qquad \text{for} \quad j = 3, \ldots, 500. \end{cases}$$

We generate 500 data matrices $A \in \mathbf{R}^{p \times n}$ and employ the four GPower algorithms as well as Greedy to compute two unit-norm sparse loading vectors $z_1, z_2 \in \mathbf{R}^{500}$, which are hoped to be close to v_1 and v_2. We consider the model underlying the data to be *successfully identified* (or *recovered*) when both quantities $|v_1^T z_1|$ and $|v_2^T z_2|$ are greater than 0.99.

Two simple alternative strategies are compared for choosing the sparsity-inducing parameters γ_1 and γ_2 required by the GPower algorithms. First, we choose them uniformly at random, between the theoretical bounds. Second, we fix them to reasonable a priori values; in particular, the midpoints of the corresponding admissible interval. For the block algorithm GPower$_{\ell_1,m}$, the parameter γ_2 is fixed at 10 percent of the corresponding upper bound. This value was chosen by limited trial and error to give good results for the particular data analyzed. We do not intend to suggest that this is a recommended choice in general. The values of the sparsity-inducing parameters for the ℓ_0-based GPower algorithms are systematically chosen as the squares of the values chosen for their ℓ_1 counterparts. More details on the selection of γ_1 and γ_2 are provided in Table 4. Concerning the parameters μ_1 and μ_2 used by the block algorithms, both situations $\mu_1 = \mu_2$ and $\mu_1 > \mu_2$ have been considered. Note that Greedy requires to specify the targeted cardinalities as an input, i.e., ten nonzeros entries for both loading vectors.

In Table 5, we provide the average of the scalar products $|z_1^T z_2|$, $|v_1^T z_1|$ and $|v_2^T z_2|$ for 500 data matrices with the covariance matrix Σ. The proportion of successful identification of the vectors v_1 and v_2 is also given. The table shows that the GPower algorithms are robust with respect to the choice of the sparsity inducing parameters γ. Good values of γ_1 and γ_2 are easily found by trial and error. The chances of recovery of the sparse model underlying the data are rather good, and some versions of the algorithms successfully recover the sparse model even when the parameters γ are chosen at random. The GPower algorithms do not appear to be as successful as Greedy, which managed to correctly identify vectors v_1 and v_2 in all tests. Note that while the latter method requires the exact knowledge of the cardinality of each component, the GPower algorithms find the sparse model that fits the data best without this information. This property of the GPower algorithms is valuable in real-data settings, where little or nothing is known a priori about the cardinality of the components.

Looking at the values reported in Table 5, we observe that the block GPower algorithms are more likely to obtain loading vectors that are “more orthogonal” when using parameters μ_j which are distinct.

Algorithm	Random	Fixed
GPower$_{\ell_1}$	γ_1 uniform distrib. on $[0, \max_i \|a_i\|_2]$	$\gamma_1 = \frac{1}{2} \max_i \|a_i\|_2$
	γ_2 uniform distrib. on $[0, \max_i \|\bar{a}_i\|_2]$	$\gamma_2 = \frac{1}{2} \max_i \|\bar{a}_i\|_2$
GPower$_{\ell_0}$	$\sqrt{\gamma_1}$ uniform distrib. on $[0, \max_i \|a_i\|_2]$	$\gamma_1 = \frac{1}{4} \max_i \|a_i\|_2^2$
	$\sqrt{\gamma_2}$ uniform distrib. on $[0, \max_i \|\bar{a}_i\|_2]$	$\gamma_2 = \frac{1}{4} \max_i \|\bar{a}_i\|_2^2$
GPower$_{\ell_1,m}$	γ_1 uniform distrib. on $[0, \max_i \|a_i\|_2]$	$\gamma_1 = \frac{1}{2} \max_i \|a_i\|_2$
with $\mu_1 = \mu_2 = 1$	γ_2 uniform distrib. on $[0, \max_i \|a_i\|_2]$	$\gamma_2 = \frac{1}{10} \max_i \|a_i\|_2$
GPower$_{\ell_0,m}$	$\sqrt{\gamma_1}$ uniform distrib. on $[0, \max_i \|a_i\|_2]$	$\gamma_1 = \frac{1}{4} \max_i \|a_i\|_2^2$
with $\mu_1 = \mu_2 = 1$	$\sqrt{\gamma_2}$ uniform distrib. on $[0, \max_i \|a_i\|_2]$	$\gamma_2 = \frac{1}{100} \max_i \|a_i\|_2^2$
GPower$_{\ell_1,m}$	γ_1 uniform distrib. on $[0, \max_i \|a_i\|_2]$	$\gamma_1 = \frac{1}{2} \max_i \|a_i\|_2$
with $\mu_1 = 1$ and $\mu_2 = 0.5$	γ_2 uniform distrib. on $[0, \frac{1}{2} \max_i \|a_i\|_2]$	$\gamma_2 = \frac{1}{20} \max_i \|a_i\|_2$
GPower$_{\ell_0,m}$	$\sqrt{\gamma_1}$ uniform distrib. on $[0, \max_i \|a_i\|_2]$	$\gamma_1 = \frac{1}{4} \max_i \|a_i\|_2^2$
with $\mu_1 = 1$ and $\mu_2 = 0.5$	$\sqrt{\gamma_2}$ uniform distrib. on $[0, \frac{1}{2} \max_i \|a_i\|_2]$	$\gamma_2 = \frac{1}{400} \max_i \|a_i\|_2^2$

Table 4: Details on the random and fixed choices of the sparsity-inducing parameters γ_1 and γ_2 leading to the results displayed in Table 5. Matrix $\bar{A}$ used in the case of the single-unit algorithms denotes the residual matrix after one deflation step.

Algorithm	γ	$\|z_1^T z_2\|$	$\|v_1^T z_1\|$	$\|v_2^T z_2\|$	Chance of success
GPower$_{\ell_1}$	random	$15.8\ 10^{-3}$	0.9693	0.9042	0.71
GPower$_{\ell_0}$	random	$15.7\ 10^{-3}$	0.9612	0.8990	0.69
GPower$_{\ell_1,m}$ with $\mu_1 = \mu_2 = 1$	random	$10.1\ 10^{-3}$	0.8370	0.2855	0.06
GPower$_{\ell_0,m}$ with $\mu_1 = \mu_2 = 1$	random	$9.2\ 10^{-3}$	0.8345	0.3109	0.07
GPower$_{\ell_1,m}$ with $\mu_1 = 1$ and $\mu_2 = 0.5$	random	$1.8\ 10^{-4}$	0.8300	0.3191	0.09
GPower$_{\ell_0,m}$ with $\mu_1 = 1$ and $\mu_2 = 0.5$	random	$1.5\ 10^{-4}$	0.8501	0.3001	0.09
GPower$_{\ell_1}$	fixed	0	0.9998	0.9997	1
GPower$_{\ell_0}$	fixed	0	0.9998	0.9997	1
GPower$_{\ell_1,m}$ with $\mu_1 = \mu_2 = 1$	fixed	$4.25\ 10^{-2}$	0.9636	0.8114	0.63
GPower$_{\ell_0,m}$ with $\mu_1 = \mu_2 = 1$	fixed	$3.77\ 10^{-2}$	0.9663	0.7990	0.67
GPower$_{\ell_1,m}$ with $\mu_1 = 1$ and $\mu_2 = 0.5$	fixed	$1.8\ 10^{-3}$	0.9875	0.9548	0.89
GPower$_{\ell_0,m}$ with $\mu_1 = 1$ and $\mu_2 = 0.5$	fixed	$6.7\ 10^{-5}$	0.9937	0.9654	0.96
PCA	–	0	0.9110	0.9063	0
Greedy	–	0	0.9998	0.9997	1

Table 5: Average of the quantities $|z_1^T z_2|$, $|v_1^T z_1|$, $|v_2^T z_2|$ and proportion of successful identifications of the two dominant sparse eigenvectors of Σ by extracting two sparse principal components from 500 data matrices. The Greedy algorithm requires prior knowledge of the cardinalities of each component, while the GPower algorithms are very likely to identify the underlying sparse model without this information.

Table 5 does not include results for the algorithms rSVD and sPCA because of our limited experience with those algorithms and the absence of such experiments in the literature but we expect a similar flexibility to the sparsity parameter tuning in view of the connections developed in Section 4.5.

5.2 Random data without underlying sparse PCA model

All random data matrices $A \in \mathbf{R}^{p \times n}$ considered in this section are generated according to a Gaussian distribution, with zero mean and unit variance.

Trade-off curves. Let us first compare the single-unit algorithms, which provide a unit-norm sparse loading vector $z \in \mathbf{R}^n$. We first plot the variance explained by the extracted component against the cardinality of the resulting loading vector z. For each algorithm, the sparsity-inducing parameter is incrementally increased to obtain loading vectors z with a cardinality that decreases from n to 1. The results displayed in Figure 1 are averages of computations on 100 random matrices with dimensions $p = 100$ and $n = 300$. The considered sparse PCA methods aggregate in two groups: GPower$_{\ell_1}$, GPower$_{\ell_0}$, Greedy and rSVD$_{\ell_0}$ outperform the SPCA and the rSVD$_{\ell_1}$ approaches. It seems that these latter methods perform worse because of the ℓ_1 penalty term used in them. If one, however, post-processes the active part of z according to (41), as we do in GPower$_{\ell_1}$, all sparse PCA methods reach the same performance.

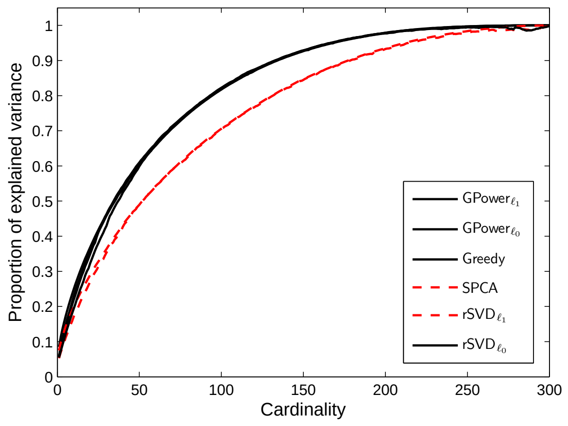

Figure 1: **Trade-off curve** between explained variance and cardinality. The vertical axis is the ratio $\mathrm{Var}(z_{\mathrm{sPCA}})/\,\mathrm{Var}(z_{\mathrm{PCA}})$, where the loading vector z_{sPCA} is computed by sparse PCA and z_{PCA} is the first principal loading vector. The considered algorithms aggregate in two groups: GPower$_{\ell_1}$, GPower$_{\ell_0}$, Greedy and rSVD$_{\ell_0}$ (top curve), and SPCA and rSVD$_{\ell_1}$ (bottom curve). For a fixed cardinality value, the methods of the first group explain more variance. Postprocessing algorithms SPCA and rSVD$_{\ell_1}$ with equation (41), results, however, in the same performance as the other algorithms.

Controlling sparsity with γ. Among the considered methods, the greedy approach is the only one to directly control the cardinality of the solution, i.e., the desired cardinality is an input of the algorithm. The other methods require a parameter controlling the trade-off between variance and cardinality. Increasing this parameter leads to solutions with smaller cardinality, but the resulting number of nonzero elements can not be precisely predicted. In Figure 2, we plot the average relationship between the parameter γ and the resulting cardinality of the loading vector z for the two algorithms GPower$_{\ell_1}$ and GPower$_{\ell_0}$. In view of (10) (resp. (15)), the entries i of the loading vector z obtained by the GPower$_{\ell_1}$ algorithm (resp. the GPower$_{\ell_0}$ algorithm) satisfying

$$\|a_i\|_2 \leq \gamma \quad \text{(resp. } \|a_i\|_2^2 \leq \gamma) \tag{43}$$

have to be zero. Taking into account the distribution of the norms of the columns of A, this provides for every γ a theoretical upper bound on the expected cardinality of the resulting vector z.

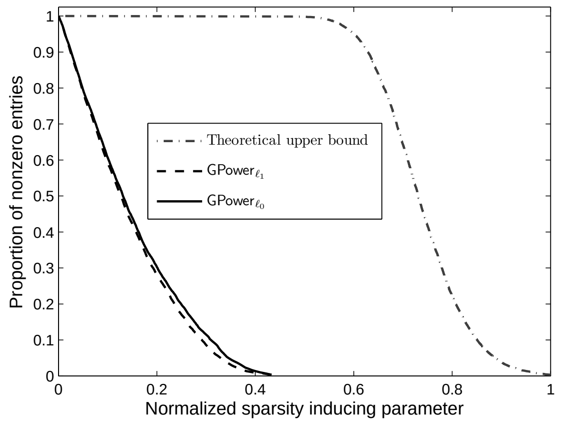

Figure 2: Dependence of cardinality on the value of the sparsity-inducing parameter γ. In case of the GPower$_{\ell_1}$ algorithm, the horizontal axis shows $\gamma/\|a_{i^*}\|_2$, whereas for the GPower$_{\ell_0}$ algorithm, we use $\sqrt{\gamma}/\|a_{i^*}\|_2$. The theoretical upper bound is therefor identical for both methods. The plots are averages based on 100 test problems of size $p = 100$ and $n = 300$.

Greedy versus the rest. From the experiments reported above, Greedy and the GPower methods appear to have similar performance in terms of quality of the obtained solution. Moreover, Greedy computes a full path of solutions up to a chosen cardinality, and does not have to deal with the issue of tuning the sparsity parameter γ. The price of this significant advantage of Greedy is its heavy computational load. In order to compare the empirical computational complexities of different algorithms, we display in Figure 3 the average time required to extract one sparse component from Gaussian matrices of dimensions $p = 100$ and $n = 300$. One immediately notices that the greedy method slows down significantly as cardinality increases, whereas the speed of the other considered algorithms does not depend on cardinality. Since on average Greedy is much slower than the other methods, even for low cardinalities, and because we aim at large-scale applications where the computational load of Greedy would be prohibitive, we discard it from the following numerical experiments.

Speed and scaling test. In Tables 6 and 7 we compare the speed of the remaining algorithms. Table 6 deals with problems with a fixed aspect ratio $n/p = 10$, whereas in Table 7, p is fixed at 500, and exponentially increasing values of n are considered. For the GPower$_{\ell_1}$ method, the sparsity inducing parameter γ was set to 10% of the upper bound $\gamma_{\max} = \|a_{i^*}\|_2$. For the GPower$_{\ell_0}$ method, γ was set to 1% of $\gamma_{\max} = \|a_{i^*}\|_2^2$ in order to aim for solutions of comparable cardinalities (see (43)). These two parameters have also been used for the rSVD$_{\ell_1}$ and the rSVD$_{\ell_0}$ methods, respectively. Concerning SPCA, the sparsity parameter has been chosen by trial and error to get, on average, solutions with similar cardinalities as obtained by the other methods. The values displayed in Tables 6 and 7 correspond to the average running times of the algorithms on 100 test instances for each problem size. In both tables, the new methods GPower$_{\ell_1}$ and GPower$_{\ell_0}$ are the fastest. The difference in speed between GPower$_{\ell_1}$ and GPower$_{\ell_0}$ results from different approaches to fill the active part of z: GPower$_{\ell_1}$ requires to compute a rank-one approximation of a submatrix of A (see Equation (41)), whereas the explicit solution (13) is available to GPower$_{\ell_0}$. The linear complexity of the algorithms in the problem size n is clearly visible in Table 7.

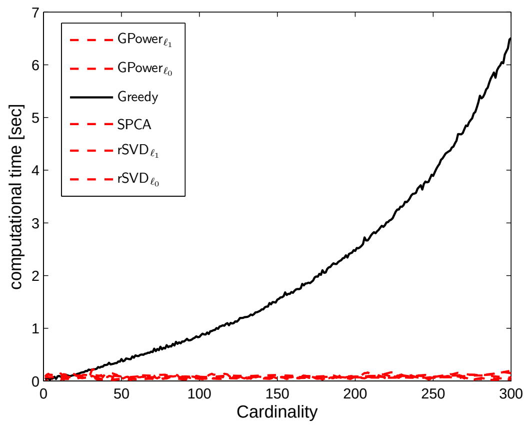

Figure 3: The computational complexity of Greedy grows significantly if it is set out to output a loading vector of increasing cardinality. The speed of the other methods is unaffected by the cardinality target.

$p \times n$	100×1000	250×2500	500×5000	750×7500	1000×10000
GPower$_{\ell_1}$	0.10	0.86	2.45	4.28	5.86
GPower$_{\ell_0}$	0.03	0.42	1.21	2.07	2.85
SPCA	0.24	2.92	14.5	40.7	82.2
rSVD$_{\ell_1}$	0.19	2.42	3.97	7.51	9.59
rSVD$_{\ell_0}$	0.18	2.14	3.85	6.94	8.34

Table 6: Average computational time for the extraction of one component (in seconds).

$p \times n$	500×1000	500×2000	500×4000	500×8000	500×16000
GPower$_{\ell_1}$	0.42	0.92	2.00	4.00	8.54
GPower$_{\ell_0}$	0.18	0.42	0.96	2.14	4.55
SPCA	5.20	7.20	12.0	22.6	44.7
rSVD$_{\ell_1}$	1.05	2.12	3.63	7.43	14.4
rSVD$_{\ell_0}$	1.02	1.97	3.45	6.58	13.2

Table 7: Average computational time for the extraction of one component (in seconds).

Different convergence mechanisms. Figure 4 illustrates how the trade-off between explained variance and sparsity evolves in the time of computation for the two methods GPower$_{\ell_1}$ and rSVD$_{\ell_1}$. In case of the GPower$_{\ell_1}$ algorithm, the initialization point (39) provides a good approximation of the final cardinality. This method then works on maximizing the variance while keeping the sparsity at a low level throughout. The rSVD$_{\ell_1}$ algorithm, in contrast, works in two steps. First, it maximizes the variance, without enforcing sparsity. This corresponds to computing the first principal component and requires thus a first run of the algorithm with random initialization and a sparsity inducing parameter set at zero. In the second run, this parameter is set to a positive value and the method works to rapidly decrease cardinality at the expense of only a modest decrease in explained variance. So, the new algorithm GPower$_{\ell_1}$ performs faster primarily because it combines the two phases into one, simultaneously optimizing the trade-off between variance and sparsity.

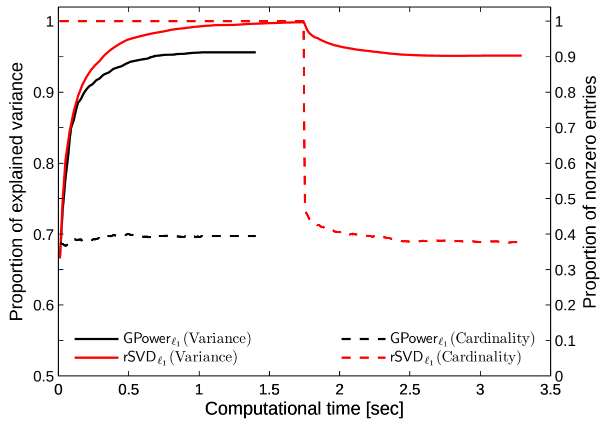

Figure 4: Evolution of the variance (solid lines and left axis) and cardinality (dashed lines and right axis) in time of computation for the methods GPower$_{\ell_1}$ and rSVD$_{\ell_1}$ on a test problem with $p = 250$ and $n = 2500$. The vertical axis is the ratio $\mathrm{Var}(z_{\mathrm{sPCA}})/\mathrm{Var}(z_{\mathrm{PCA}})$, where the loading vector z_{sPCA} is computed by sparse PCA and z_{PCA} is the first principal loading vector. The rSVD$_{\ell_1}$ algorithm first solves unconstrained PCA, whereas GPower$_{\ell_1}$ immediately optimizes the trade-off between variance and sparsity.

Extracting more components. Similar numerical experiments, which include the methods GPower$_{\ell_1,m}$ and GPower$_{\ell_0,m}$, have been conducted for the extraction of more than one component. A deflation scheme is used by the non-block methods to sequentially compute m components. These experiments lead to similar conclusions as in the single-unit case, i.e, the methods GPower$_{\ell_1}$, GPower$_{\ell_0}$, GPower$_{\ell_1,m}$, GPower$_{\ell_0,m}$ and rSVD$_{\ell_0}$ outperform the SPCA and rSVD$_{\ell_1}$ approaches in terms of variance explained at a fixed cardinality. Again, these last two methods can be improved by postprocessing the resulting loading vectors with Algorithm 6, as it is done for GPower$_{\ell_1,m}$. The average running times for problems of various sizes are listed in Table 8. The new power-like methods are significantly faster on all instances.

$p \times n$	50×500	100×1000	250×2500	500×5000	750×7500
GPower$_{\ell_1}$	0.22	0.56	4.62	12.6	20.4
GPower$_{\ell_0}$	0.06	0.17	2.15	6.16	10.3
GPower$_{\ell_1,m}$	0.09	0.28	3.50	12.4	23.0
GPower$_{\ell_0,m}$	0.05	0.14	2.39	7.7	12.4
SPCA	0.61	1.47	13.4	48.3	113.3
rSVD$_{\ell_1}$	0.29	1.12	7.72	22.6	46.1
rSVD$_{\ell_0}$	0.28	1.03	7.21	20.7	41.2

Table 8: Average computational time for the extraction of $m = 5$ components (in seconds).

Cost and benefits of the post-processing phase. Figure 5 illustrates the evolution of the relative increase of computational time as well as the relative improvement in terms of explained variance due to the post-processing phase for increasing values of γ. Only the methods with iterative post-processing algorithms are considered, i.e., GPower$_{\ell_1}$ (left-hand plot) and GPower$_{\ell_1,m}$ (right-hand plot). In the single unit case, the post-processing phase, which amounts to a rank-one

SVD of the truncated data matrix A_P, becomes less costly as the level of sparsity increases. As expected, the improvement of variance increases when γ gets larger, i.e., when the ℓ_1-penalty biases more and more the values assigned to the non-zero entries of the vector z. A similar observation holds in the block case, excepted that the relative excess of computational time took by the post-processing increases with γ. This difference with the single-unit case results from the fact that the post-processing in the block case deals with sparse matrices of possibly large dimension, whereas in the single-unit case the problem is easily rewritten in terms of a full vector with a dimension that equals the number of nonzero elements. Overall, the postprocessing uses less that 10% of the time needed by the main routine, to improve the explained variance by up to 30%.

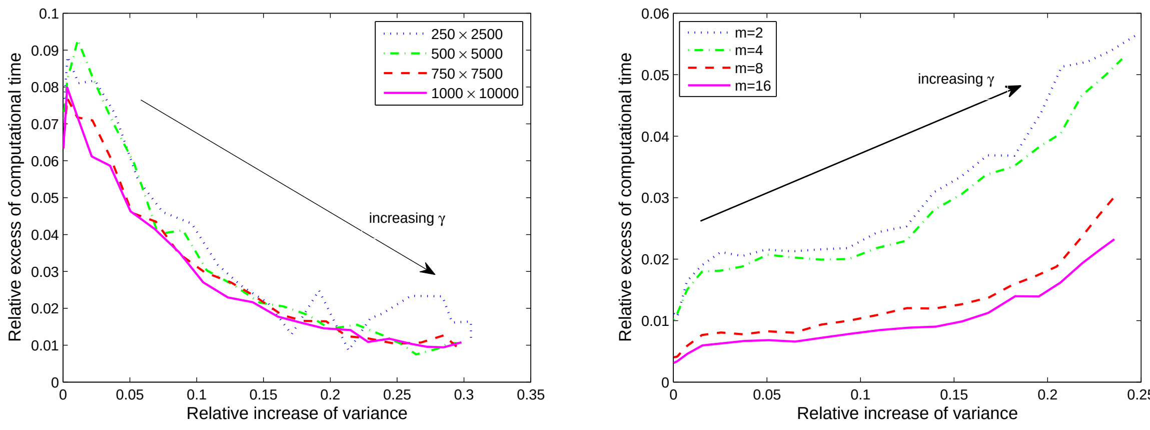

Figure 5: Effects of post-processing in the case of the algorithms GPower$_{\ell_1}$ (left-hand plot) and GPower$_{\ell_1,m}$ (right-hand plot) for increasing values of γ. The horizontal axis is the quantity $\mathrm{Var}_{\text{after post-processing}} / \mathrm{Var}_{\text{before post-processing}} - 1$ and the vertical axis is the ratio CPU time$_{\text{for the postprocessing}}$/CPU time$_{\text{for the main routine}}$. For GPower$_{\ell_1}$, several problem sizes are considered, whereas the curves for GPower$_{\ell_1,m}$ relate to matrices of dimension 500-by-5000, but for several numbers m of extracted components. Each curve is an average on 25 random Gaussian data matrices.

5.3 Pitprops data

The "pitprops" data, which stores 180 observations of 13 variables, has been a standard benchmark to evaluate algorithms for sparse PCA (see e.g., Jolliffe et al. (2003); Zou et al. (2006); Moghaddam et al. (2006); Shen and Huang (2008)). Following these previous studies, we use the GPower algorithms to compute *six* sparse principal components of the data. For such more-samples-than-variables settings, it is customary to first factor the covariance matrix as $\Sigma = A^T A$ with $A \in \mathbf{R}^{13\times 13}$, such that the dimension p is virtually reduced to 13. This operation can be readily done through the eigenvalue decomposition of Σ.

In Table 9, we provide the total cardinality and the proportion of adjusted variance explained by six components computed with SPCA, rSVD$_{\ell_1}$, Greedy as well as our GPower algorithms. The results concerning SPCA, rSVD$_{\ell_1}$, Greedy correspond to the patterns of zeros and nonzeros proposed by Zou et al. (2006), Shen and Huang (2008) and Moghaddam et al. (2006), respectively. For fair comparison, the pattern related to SPCA and rSVD$_{\ell_1}$ have been post-processed with the approach proposed in Section 4.2. Concerning the Gpower algorithms, we fix the six parameters γ_j at the same ratio of their respective upper-bounds. For the block algorithm $\text{GPower}_{\ell_1,m}$, experiments have been conducted in both cases "identical μ_j" and "distinct μ_j".

Table 9 illustrates that better patterns can be identified with the GPower algorithms, i.e., patterns that explain more variance with the same cardinality (and sometimes even with a smaller one). These results are furthermore likely to be improved by a fine tuning of the six parameters γ_j (i.e., by choosing them independently from each others).

Method	Parameters	Total cardinality	Prop. of explained variance
rSVD_{ℓ_1}	see Shen and Huang (2008)	25	0.7924
SPCA	see Zou et al. (2006)	18	0.7680
Greedy	cardinalities: 6-2-3-1-1-1	14	0.7150
	cardinalities: 5-2-2-1-1-1	12	0.5406
GPower_{ℓ_1}	$\gamma_j/\bar{\gamma}_j = 0.22,$ for $j = 1, \ldots, 6$	25	0.8083
	$\gamma_j/\bar{\gamma}_j = 0.28$	18	0.7674
	$\gamma_j/\bar{\gamma}_j = 0.30$	15	0.7542
	$\gamma_j/\bar{\gamma}_j = 0.40$	13	0.7172
	$\gamma_j/\bar{\gamma}_j = 0.50$	11	0.6042
$\text{GPower}_{\ell_1,m}$	$\gamma_j/\bar{\gamma}_j = 0.17,$ for $j = 1, \ldots, 6$	25	0.7733
with $\mu_j = 1$	$\gamma_j/\bar{\gamma}_j = 0.25$	17	0.7708
	$\gamma_j/\bar{\gamma}_j = 0.3$	14	0.7508
	$\gamma_j/\bar{\gamma}_j = 0.4$	13	0.7076
	$\gamma_j/\bar{\gamma}_j = 0.45$	11	0.6603
$\text{GPower}_{\ell_1,m}$	$\gamma_j/\bar{\gamma}_j = 0.18,$ for $j = 1, \ldots, 6$	25	0.8111
	$\gamma_j/\bar{\gamma}_j = 0.25$	18	0.7849
with $\mu_j = \frac{1}{j}$	$\gamma_j/\bar{\gamma}_j = 0.30$	15	0.7610
	$\gamma_j/\bar{\gamma}_j = 0.35$	13	0.7323
	$\gamma_j/\bar{\gamma}_j = 0.40$	12	0.6656

Table 9: Extraction of 6 components from the pitprops data. For GPower_{ℓ_1}, one defines the upper-bounds $\bar{\gamma}_j = \max_i \|a_i^{(j)}\|_2$, where $A^{(j)}$ is the residual data matrix after $j-1$ deflation steps. For $\text{GPower}_{\ell_1,m}$, the upper-bounds are $\bar{\gamma}_j = \mu_j \max_i \|a_i\|_2$.

5.4 Analysis of gene expression data

Gene expression data results from DNA microarrays and provide the expression level of thousands of genes across several hundreds of experiments. The interpretation of these huge databases remains a challenge. Of particular interest is the identification of genes that are systematically coexpressed under similar experimental conditions. We refer to Riva et al. (2005) and references therein for more details on microarrays and gene expression data. PCA has been intensively applied in this context (e.g., Alter et al. (2003)). Further methods for dimension reduction, such as independent component analysis (Liebermeister (2002)) or nonnegative matrix factorization (Brunet et al. (2004)), have also been used on gene expression data. Sparse PCA, which extracts components involving a few genes only, is expected to enhance interpretation.

Data sets. The results below focus on four major data sets related to breast cancer. They are briefly detailed in Table 10.[7] Each sparse PCA algorithm computes ten components from these data sets, i.e., $m = 10$.

Study	Samples (p)	Genes (n)	Reference
Vijver	295	13319	van de Vijver et al. (2002)
Wang	285	14913	Wang et al. (2005)
Naderi	135	8278	Naderi et al. (2007)
JRH-2	101	14223	Sotiriou et al. (2006)

Table 10: Breast cancer cohorts.

Speed. The average computational time required by the sparse PCA algorithms on each data set is displayed in Table 11. The indicated times are averages on all the computations performed to obtain cardinality ranging from n down to 1.

	Vijver	Wang	Naderi	JRH-2
GPower_{ℓ_1}	5.92	5.33	2.15	2.69
GPower_{ℓ_0}	4.86	4.93	1.33	1.73
$\mathsf{GPower}_{\ell_1,m}$	5.40	4.37	1.77	1.14
$\mathsf{GPower}_{\ell_0,m}$	5.61	7.21	2.25	1.47
SPCA	77.7	82.1	26.7	11.2
rSVD_{ℓ_1}	10.19	9.97	3.96	4.43
rSVD_{ℓ_0}	9.51	9.23	3.46	3.61

Table 11: Average computational times (in seconds) for the extraction of $m = 10$ components.

Trade-off curves. Figure 6 plots the proportion of adjusted variance versus the cardinality for the "Vijver" data set. The other data sets have similar plots. As for the random test problems, this performance criterion does not discriminate among the different algorithms. All methods have in fact the same performance, provided that the SPCA and rSVD_{ℓ_1} approaches are used with postprocessing by Algorithm 6.

Interpretability. A more interesting performance criterion is to estimate the biological interpretability of the extracted components. The *pathway enrichment index* (PEI) proposed by Teschendorff et al. (2007) measures the statistical significance of the overlap between two kinds of gene sets. The first sets are inferred from the computed components by retaining the most expressed genes, whereas the second sets result from biological knowledge. For instance, metabolic pathways provide sets of genes known to participate together when a certain biological function is required. An alternative is given by the regulatory motifs: genes tagged with an identical motif are likely to be coexpressed. One expects sparse PCA methods to recover some of these biologically significant sets. Table 12 displays the PEI based on 536 metabolic pathways related to cancer. The PEI is the fraction of these 536 sets presenting a statistically significant overlap with the genes inferred from the sparse principal components. The values in Table 12 correspond to the largest PEI obtained

7. The normalized data sets have been kindly provided by Andrew Teschendorff.

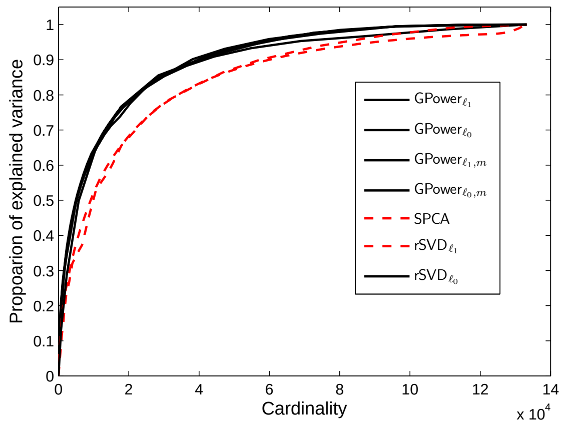

Figure 6: Evolution of the explained variance with the cardinality (case of the “Vijver” data). The y-axis is the ratio $\mathrm{AdjVar}(Z_{\mathrm{sPCA}})/\,\mathrm{AdjVar}(Z_{\mathrm{PCA}})$ where the loading vectors Z_{sPCA} are computed by sparse PCA and Z_{PCA} are the m first principal loading vectors.

among all possible cardinalities. Similarly, Table 13 is based on 173 motifs. More details on the selected pathways and motifs can be found in Teschendorff et al. (2007). This analysis clearly indicates that the sparse PCA methods perform much better than PCA in this context. Furthermore, the new GPower algorithms, and especially the block formulations, provide largest PEI values for both types of biological information. In terms of biological interpretability, they systematically outperform previously published algorithms.

	Vijver	Wang	Naderi	JRH-2
PCA	0.0728	0.0466	0.0149	0.0690
GPower$_{\ell_1}$	**0.1493**	0.1026	0.0728	0.1250
GPower$_{\ell_1}$	0.1250	0.1250	0.0672	0.1026
GPower$_{\ell_1,m}$	0.1418	0.1250	**0.1026**	**0.1381**
GPower$_{\ell_0,m}$	0.1362	**0.1287**	0.1007	0.1250
SPCA	0.1362	0.1007	0.0840	0.1007
rSVD$_{\ell_1}$	0.1213	0.1175	0.0914	0.0914
rSVD$_{\ell_0}$	0.1175	0.0970	0.0634	0.1063

Table 12: PEI-values based on a set of 536 cancer-related pathways.

6. Conclusion

We have proposed two single-unit and two block formulations of the sparse PCA problem and constructed reformulations with several favorable properties. First, the reformulated problems are of the form of maximization of a convex function on a compact set, with the feasible set being either a unit Euclidean sphere or the Stiefel manifold. This structure allows for the design and iteration complexity analysis of a simple gradient scheme which applied to our sparse PCA setting results in four new algorithms for computing sparse principal components of a matrix $A \in \mathbf{R}^{p\times n}$. Second, our algorithms appear to be faster if either the objective function or the feasible set are strongly convex,

	Vijver	Wang	Naderi	JRH-2
PCA	0.0347	0	0.0289	0.0405
GPower$_{\ell_1}$	0.1850	0.0867	0.0983	0.1792
GPower$_{\ell_0}$	0.1676	0.0809	0.0925	**0.1908**
GPower$_{\ell_1,m}$	**0.1908**	**0.1156**	**0.1329**	0.1850
GPower$_{\ell_0,m}$	0.1850	0.1098	**0.1329**	0.1734
SPCA	0.1734	0.0925	0.0809	0.1214
rSVD$_{\ell_1}$	0.1387	0.0809	0.1214	0.1503
rSVD$_{\ell_0}$	0.1445	0.0867	0.0867	0.1850

Table 13: PEI-values based on a set of 173 motif-regulatory gene sets.

which holds in the single-unit case and can be enforced in the block case. Third, the dimension of the feasible sets does not depend on n but on p and on the number m of components to be extracted. This is a highly desirable property if $p \ll n$. Last but not least, on random and real-life biological data, our methods systematically outperform the existing algorithms both in speed and trade-off performance. Finally, in the case of the biological data, the components obtained by our block algorithms deliver the richest biological interpretation as compared to the components extracted by the other methods.

Acknowledgments

This paper presents research results of the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office. The scientific responsibility rests with its authors. Research of Yurii Nesterov and Peter Richtárik has been supported by the grant “Action de recherche concertée ARC 04/09-315” from the “Direction de la recherche scientifique - Communauté française de Belgique”. Michel Journée is a research fellow of the Belgian National Fund for Scientific Research (FNRS).

7. Appendix A

In this appendix we characterize a class of functions with strongly convex level sets. First we need to collect some basic preliminary facts. All the inequalities of Proposition 11 are well-known in the literature.

Proposition 11 *(i) If f is a strongly convex function with convexity parameter σ_f, then for all x, y and $0 \le \alpha \le 1$,*

$$f(\alpha x + (1-\alpha)y) \le \alpha f(x) + (1-\alpha) f(y) - \frac{\sigma_f}{2}\alpha(1-\alpha)\|x-y\|^2. \tag{44}$$

(ii) If f is a convex differentiable function and its gradient is Lipschitz continuous with constant L_f, then for all x and h,

$$f(x+h) \le f(x) + \langle f'(x), h\rangle + \frac{L_f}{2}\|h\|^2, \tag{45}$$

and

$$\|f'(x)\|_* \leq \sqrt{2L_f(f(x) - f_*)}, \tag{46}$$

where $f_* \overset{def}{=} \min_{x\in\mathbf{E}} f(x)$.

We are now ready for the main result of this section.

Theorem 12 (Strongly convex level sets) *Let $f : \mathbf{E} \to \mathbf{R}$ be a nonnegative strongly convex function with convexity parameter $\sigma_f > 0$. Also assume f has a Lipschitz continuous gradient with Lipschitz constant $L_f > 0$. Then for any $\omega > 0$, the set*

$$\mathcal{Q}_\omega \overset{def}{=} \{x \mid f(x) \leq \omega\}$$

is strongly convex with convexity parameter

$$\sigma_{\mathcal{Q}_\omega} = \frac{\sigma_f}{\sqrt{2\omega L_f}}.$$

Proof Consider any $x, y \in \mathcal{Q}_\omega$, scalar $0 \leq \alpha \leq 1$ and let $z_\alpha = \alpha x + (1-\alpha)y$. Notice that by convexity, $f(z_\alpha) \leq \omega$. For any $u \in \mathbf{E}$,

$$\begin{aligned}
f(z_\alpha + u) &\overset{(45)}{\leq} f(z_\alpha) + \langle f'(z_\alpha), u\rangle + \frac{L_f}{2}\|u\|^2 \\
&\leq f(z_\alpha) + \|f'(z_\alpha)\|\|u\| + \frac{L_f}{2}\|u\|^2 \\
&\overset{(46)}{\leq} f(z_\alpha) + \sqrt{2L_f f(z_\alpha)}\|u\| + \frac{L_f}{2}\|u\|^2 \\
&= \left(\sqrt{f(z_\alpha)} + \sqrt{\tfrac{L_f}{2}}\|u\|\right)^2 \\
&\overset{(44)}{\leq} \left(\sqrt{\omega - \beta} + \sqrt{\tfrac{L_f}{2}}\|u\|\right)^2,
\end{aligned}$$

where

$$\beta = \frac{\sigma_f}{2}\alpha(1-\alpha)\|x-y\|^2. \tag{47}$$

In view of (30), it remains to show that the last displayed expression is bounded above by ω whenever u is of the form

$$u = \frac{\sigma_{\mathcal{Q}_\omega}}{2}\alpha(1-\alpha)\|x-y\|^2 s = \frac{\sigma_f}{2\sqrt{2\omega L_f}}\alpha(1-\alpha)\|x-y\|^2 s, \tag{48}$$

for some $s \in \mathcal{S}$. However, this follows directly from concavity of the scalar function $g(t) = \sqrt{t}$:

$$\begin{aligned}
\sqrt{\omega - \beta} = g(\omega - \beta) &\leq g(\omega) - \langle g'(\omega), \beta\rangle \\
&= \sqrt{\omega} - \frac{\beta}{2\sqrt{\omega}} \\
&\overset{(47)}{\leq} \sqrt{\omega} - \frac{\sigma_f}{4\sqrt{\omega}}\alpha(1-\alpha)\|x-y\|^2 \\
&\overset{(48)}{\leq} \sqrt{\omega} - \sqrt{\frac{L_f}{2}}\|u\|.
\end{aligned}$$

■

Example 13 *Let $f(x) = \|x\|^2$. Note that $\sigma_f = L_f = 2$. If we let $\omega = r^2$, then*

$$\mathcal{Q}_\omega = \{x \mid f(x) \leq \omega\} = \{x \mid \|x\| \leq r\} = r \cdot \mathcal{B}.$$

We have shown before (see the discussion immediately following Assumption 3), that the strong convexity parameter of this set is $\sigma_{\mathcal{Q}_\omega} = \frac{1}{r}$. Note that we recover this as a special case of Theorem 12:

$$\sigma_{\mathcal{Q}_\omega} = \frac{\sigma_f}{\sqrt{2\omega L_f}} = \frac{1}{r}.$$

References

P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, January 2008.

O. Alter, P. O. Brown, and D. Botstein. Generalized singular value decomposition for comparative analysis of genome-scale expression data sets of two different organisms. *Proc Natl Acad Sci USA*, 100(6):3351–3356, 2003.

R. W. Brockett. Dynamical systems that sort lists, diagonalize matrices and solve linear programming problems. *Linear Algebra Appl.*, 146:79–91, 1991.

J. P. Brunet, P. Tamayo, T. R. Golub, and J. P. Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *Proc Natl Acad Sci USA*, 101(12):4164–4169, 2004.

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

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*, 49:434–448, 2007.

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

C. Fraikin, Yu. Nesterov, and P. Van Dooren. A gradient-type algorithm optimizing the coupling between matrices. *Linear Algebra and its Applications*, 429(5-6):1229–1242, 2008.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996.

R. A. Horn and C. A. Johnson. *Matrix analysis*. Cambridge University Press, Cambridge, UK, 1985.

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

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(3):531–547, 2003.

W. Liebermeister. Linear modes of gene expression determined by independent component analysis. *Bioinformatics*, 18(1):51–60, 2002.

L. Mackey. Deflation methods for sparse PCA. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1017–1024, 2008.

B. Moghaddam, Y. Weiss, and S. Avidan. Spectral bounds for sparse PCA: Exact and greedy algorithms. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 915–922. MIT Press, Cambridge, MA, 2006.

A. Naderi, A. E. Teschendorff, N. L. Barbosa-Morais, S. E. Pinder, A. R. Green, D. G. Powe, J. F. R. Robertson, S. Aparicio, I. O. Ellis, J. D. Brenton, and C. Caldas. A gene expression signature to predict survival in breast cancer across independent data sets. *Oncogene*, 26:1507–1516, 2007.

Beresford N. Parlett. *The symmetric eigenvalue problem*. Prentice-Hall Inc., Englewood Cliffs, N.J., 1980. ISBN 0-13-880047-2. Prentice-Hall Series in Computational Mathematics.

A. Riva, A.-S. Carpentier, B. Torrésani, and A. Hénaut. Comments on selected fundamental aspects of microarray analysis. *Computational Biology and Chemistry*, 29(5):319–336, 2005.

Haipeng Shen and Jianhua Z. Huang. Sparse principal component analysis via regularized low rank matrix approximation. *Journal of Multivariate Analysis*, 99(6):1015–1034, 2008.

C. Sotiriou, P. Wirapati, S. Loi, A. Harris, S. Fox, J. Smeds, H. Nordgren, P. Farmer, V. Praz, B. Haibe-Kains, C. Desmedt, D. Larsimont, F. Cardoso, H. Peterse, D. Nuyten, M. Buyse, M. J. Van de Vijver, J. Bergh, M. Piccart, and M. Delorenzi. Gene expression profiling in breast cancer: understanding the molecular basis of histologic grade to improve prognosis. *J Natl Cancer Inst*, 98(4):262–272, 2006.

A. Teschendorff, M. Journée, P.-A. Absil, R. Sepulchre, and C. Caldas. Elucidating the altered transcriptional programs in breast cancer using independent component analysis. *PLoS Computational Biology*, 3(8):1539–1554, 2007.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58(2):267–288, 1996.

M. J. van de Vijver, Y. D. He, L. J. van't Veer, H. Dai, A. A. Hart, D. W. Voskuil, G. J. Schreiber, J. L. Peterse, C. Roberts, M. J. Marton, M. Parrish, D. Atsma, A. Witteveen, A. Glas, L. Delahaye, T. van der Velde, H. Bartelink, S. Rodenhuis, E. T. Rutgers, S. H. Friend, and R. Bernards. A gene-expression signature as a predictor of survival in breast cancer. *N Engl J Med*, 347(25): 1999–2009, 2002.

Y. Wang, J. G. Klijn, Y. Zhang, A. M. Sieuwerts, M. P. Look, F. Yang, D. Talantov, M. Timmermans, M. E. Meijer-van Gelder, J. Yu, T. Jatkoe, E. M. Berns, D. Atkins, and J. A. Foekens. Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer. *Lancet*, 365(9460):671–679, 2005.

H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286, 2006.