A direct formulation for sparse PCA using semidefinite programming

Alexandre d'Aspremont
EECS Dept.
U.C. Berkeley
Berkeley, CA 94720
alexandre.daspremont@m4x.org

Laurent El Ghaoui
SAC Capital
540 Madison Avenue
New York, NY 10029
laurent.elghaoui@sac.com
(on leave from EECS, U.C. Berkeley)

Michael I. Jordan
EECS and Statistics Depts.
U.C. Berkeley
Berkeley, CA 94720
jordan@cs.berkeley.edu

Gert R. G. Lanckriet
EECS Dept.
U.C. Berkeley
Berkeley, CA 94720
gert@eecs.berkeley.edu

Abstract

We examine the problem of approximating, in the Frobenius-norm sense, a positive, semidefinite symmetric matrix by a rank-one matrix, with an upper bound on the cardinality of its eigenvector. The problem arises in the decomposition of a covariance matrix into sparse factors, and has wide applications ranging from biology to finance. We use a modification of the classical variational representation of the largest eigenvalue of a symmetric matrix, where cardinality is constrained, and derive a semidefinite programming based relaxation for our problem.

1 Introduction

Principal component analysis (PCA) is a popular tool for data analysis and dimensionality reduction. It has applications throughout science and engineering. In essence, PCA finds linear combinations of the variables (the so-called principal components) that correspond to directions of maximal variance in the data. It can be performed via a singular value decomposition (SVD) of the data matrix A, or via an eigenvalue decomposition if A is a covariance matrix.

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

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

Rotation techniques are often used to improve interpretation of the standard principal components [1]. [2] considered simple principal components by restricting the loadings to take values from a small set of allowable integers, such as 0, 1, and -1. [3] propose an ad hoc way to deal with the problem, where the loadings with small absolute value are thresholded to zero. We will call this approach "simple thresholding." Later, a method called SCoTLASS was introduced by [4] to find modified principal components with possible zero loadings. In [5] a new approach, called sparse PCA (SPCA), was proposed to find modified components with zero loadings, based on the fact that PCA can be written as a regression-type optimization problem. This allows the application of LASSO [6], a penalization technique based on the L_1 norm.

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

SDP problems can be solved in polynomial time via general-purpose interior-point methods [7], and our current implementation of DSPCA makes use of these general-purpose methods. This suffices for an initial empirical study of the properties of DSPCA and for comparison to the algorithms discussed above on problems of small to medium dimensionality. For high-dimensional problems, the general-purpose methods are not viable and it is necessary to attempt to exploit special structure in the problem. It turns out that our problem can be expressed as a special type of saddle-point problem that is well suited to recent specialized algorithms, such as those described in [8, 9]. These algorithms offer a significant reduction in computational time compared to generic SDP solvers. In the current paper, however, we restrict ourselves to an investigation of the basic properties of DSPCA on problems for which the generic methods are adequate.

Our paper is structured as follows. In Section 2, we show how to efficiently derive a sparse rank-one approximation of a given matrix using a semidefinite relaxation of the sparse PCA problem. In Section 3, we derive an interesting robustness interpretation of our technique, and in Section 4 we describe how to use this interpretation in order to decompose a matrix into sparse factors. Section 5 outlines different algorithms that can be used to solve the problem, while Section 6 presents numerical experiments comparing our method with existing techniques.

Notation

Here, $\mathbf{S}^n$ is the set of symmetric matrices of size n. We denote by $\mathbf{1}$ a vector of ones, while $\mathbf{Card}(x)$ is the cardinality (number of non-zero elements) of a vector x. For $X \in \mathbf{S}^n$, $\|X\|_F$ is the Frobenius norm of X, i.e., $\|X\|_F = \sqrt{\mathbf{Tr}(X^2)}$, and by $\lambda^{\max}(X)$ the maximum eigenvalue of X, while $|X|$ is the matrix whose elements are the absolute values of the elements of X.

2 Sparse eigenvectors

In this section, we derive a semidefinite programming (SDP) relaxation for the problem of approximating a symmetric matrix by a rank one matrix with an upper bound on the cardinality of its eigenvector. We first reformulate this as a variational problem, we then obtain a lower bound on its optimal value via an SDP relaxation (we refer the reader to [10] for an overview of semidefinite programming).

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

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

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

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

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

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

where

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

To compute a semidefinite relaxation of this program (see [10], for example), we rewrite (2) as:

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

in the symmetric, matrix variable $X \in \mathbf{S}^n$. Indeed, if X is a solution to the above problem, then $X \succeq 0$ and $\mathbf{Rank}(X) = 1$ means that we have $X = xx^T$, and $\mathbf{Tr}(X) = 1$ implies that $\|x\|_2 = 1$. Finally, if $X = xx^T$ then $\mathbf{Card}(X) \leq k^2$ is equivalent to $\mathbf{Card}(x) \leq k$. Naturally, problem (3) is still non-convex and very difficult to solve, due to the rank and cardinality constraints. Since for every $u \in \mathbf{R}^p$, $\mathbf{Card}(u) = q$ implies $\|u\|_1 \leq \sqrt{q}\|u\|_2$, we can replace the non-convex constraint $\mathbf{Card}(X) \leq k^2$, by a weaker but convex one: $\mathbf{1}^T|X|\mathbf{1} \leq k$, where we have exploited the property that $\|X\|_F = \sqrt{x^T x} = 1$ when $X = xx^T$ and $\mathbf{Tr}(X) = 1$. If we also drop the rank constraint, we can form a relaxation of (3) and (2) as:

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

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

3 A robustness interpretation

In this section, we show that problem (4) can be interpreted as a robust formulation of the maximum eigenvalue problem, with additive, component-wise uncertainty in the matrix A. We again assume A to be symmetric and positive semidefinite. In the previous section, we considered in (2) a cardinality-constrained variational formulation of the maximum eigenvalue problem. Here we look at a small variation where we penalize the cardinality and solve:

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

in the variable $x \in \mathbf{R}^n$, where the parameter $\rho > 0$ controls the size of the penalty. Let us remark that we can easily move from the constrained formulation in (4) to the penalized form in (5) by duality. This problem is again non-convex and very difficult to solve. As in the last section, we can form the equivalent program:

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

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

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

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

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

and we get a dual to (5) as:

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

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

4 Sparse decomposition

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

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

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

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

and iterate to obtain further components.

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

5 Algorithms

For problems of moderate size, our SDP can be solved efficiently using solvers such as SEDUMI [7]. For larger-scale problems, we need to resort to other types of algorithms for convex optimization. Of special interest are the recently-developed algorithms due to [8, 9]. These are first-order methods specialized to problems having a specific saddle-point structure. It turns out that our problem, when expressed in the saddle-point form (6), falls precisely into this class of algorithms. Judged from the results presented in [9], in the closely related context of computing the Lovascz capacity of a graph, the theoretical complexity, as well as practical performance, of the method as applied to (6) should exhibit very significant improvements over the general-purpose interior-point algorithms for SDP. Of course, nothing comes without a price: for *fixed* problem size, the first-order methods mentioned above converge in $O(1/\epsilon)$, where ϵ is the required accuracy on the optimal value, while interior-point methods converge in $O(\log(1/\epsilon))$. We are currently evaluating the impact of this tradeoff both theoretically and in practice.

6 Numerical results

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

6.1 Controlling sparsity with k

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

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

We then form a test matrix $A = U^T U + \sigma v v^T$, where σ is a signal-to-noise ratio equal to 15 in our case. We sample 50 different matrices A using this technique. For each k between 1 and 10 and each A, we solve the following SDP in (4). We then extract the first eigenvector of the solution X and record its cardinality. In Figure 1, we show the mean cardinality (and standard deviation) as a function of k. We observe that $k + 1$ is actually a good predictor of the cardinality, especially when $k + 1$ is close to the actual cardinality (5 in this case).

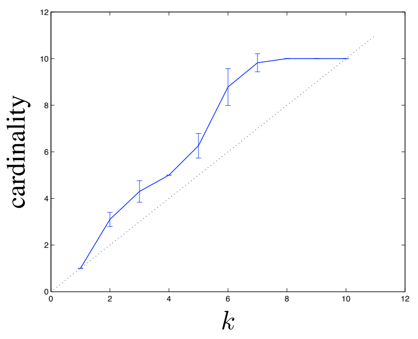

Figure 1: Cardinality versus k.

6.2 Artificial data

We consider the simulation example proposed by [5]. In this example, three hidden factors are created:

$$V_1 \sim \mathcal{N}(0, 290), \;\; V_2 \sim \mathcal{N}(0, 300), \;\; V_3 = -0.3V_1 + 0.925V_2 + \epsilon, \;\; \epsilon \sim \mathcal{N}(0, 300) \quad (8)$$

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

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

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

Since the three underlying factors have about the same variance, and the first two are associated with 4 variables while the last one is only associated with 2 variables, V_1 and V_2 are almost equally important, and they are both significantly more important than V_3. This, together with the fact that the first 2 principal components explain more than 99% of the total variance, suggests that considering two sparse linear combinations of the original variables should be sufficient to explain most of the variance in data sampled from this model. This is also discussed by [5]. The ideal solution would thus be to only use the variables (X_1, X_2, X_3, X_4) for the first sparse principal component, to recover the factor V_1, and only (X_5, X_6, X_7, X_8) for the second sparse principal component to recover V_2.

Using the true covariance matrix and the oracle knowledge that the ideal sparsity is 4, [5] performed SPCA (with $\lambda = 0$). We carry out our algorithm with $k = 4$. The results are reported in Table 1, together with results for PCA, simple thresholding and SCoTLASS ($t = 2$). Notice that SPCA, DSPCA and SCoTLASS all find the correct sparse principal components, while simple thresholding yields inferior performance. The latter wrongly includes the variables X_9 and X_{10} to explain most variance (probably it gets misled by the high correlation between V_2 and V_3), even more, it assigns higher loadings to X_9 and X_{10} than to one of the variables (X_5, X_6, X_7, X_8) that are clearly more important. Simple thresholding correctly identifies the second sparse principal component, probably because V_1 has a lower correlation with V_3. Simple thresholding also explains a bit less variance than the other methods.

6.3 Pit props data

The pit props data (consisting of 180 observations and 13 measured variables) was introduced by [11] and has become a standard example of the potential difficulty in interpreting

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

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

principal components. [4] applied SCoTLASS to this problem and [5] used their SPCA approach, both with the goal of obtaining sparse principal components that can better be interpreted than those of PCA. SPCA performs better than SCoTLASS: it identifies principal components with respectively 7, 4, 4, 1, 1, and 1 non-zero loadings, as shown in Table 2. As shown in [5], this is much sparser than the modified principal components by SCoTCLASS, while explaining nearly the same variance (75.8% versus 78.2% for the 6 first principal components). Also, simple thresholding of PCA, with a number of non-zero loadings that matches the result of SPCA, does worse than SPCA in terms of explained variance.

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

Table 2: Loadings for first three principal components, for the real-life example.

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

7 Conclusion

The semidefinite relaxation of the sparse principal component analysis problem proposed here appears to significantly improve the solution's sparsity, while explaining the same

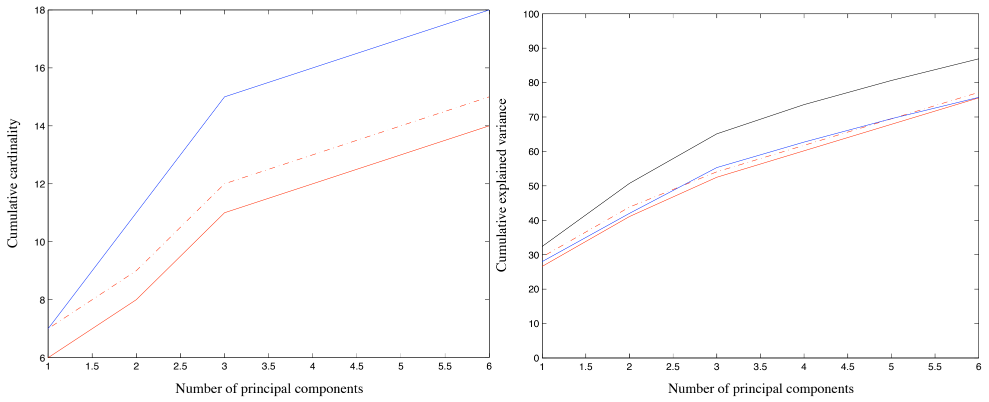

Figure 2: Cumulative cardinality and cumulative explained variance for SPCA and DSPCA as a function of the number of principal components: black line for normal PCA, blue for SPCA and red for DSPCA (full for $k_1 = 5$ and dash-dot for $k_1 = 6$).

variance as previously proposed methods in the examples detailed above. The algorithms we used here handle moderate size problems efficiently. We are currently working on large-scale extensions using first-order techniques.

Acknowledgements

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

References

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

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

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

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

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

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

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

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

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

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

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