A direct formulation for sparse PCA using semidefinite programming

A. d'Aspremont, L. El Ghaoui, M. Jordan, G. Lanckriet

ORFE, Princeton University & EECS, U.C. Berkeley

Introduction

PCA is a classic tool in multivariate data analysis

- Input: a covariance matrix A
- Output: a sequence of *factors* ranked by *variance*
- Each factor is a *linear* combination of the problem variables

Typical use: *reduce the number of dimensions* of a model while *maximizing the information* (variance) contained in the simplified model.

Introduction

Numerically: just an eigenvalue decomposition of the covariance matrix:

$$A = \sum_{i=1}^{n} \lambda_i x_i x_i^T$$

where. . .

- The factors x_i are uncorrelated
- The result of the PCA is usually not sparse, i.e. each factor is a linear combination of *all the variables* in the model.

Can we get *sparse* factors instead?

Applications, previous works

Why *sparse* factors?

- Financial time series analysis, dimensionality reduction, hedging, etc (Rebonato (1998),...)
- Multiscale data processing (Chennubhotla & Jepson (2001),...)
- Gene expression data (survey by Wall, Rechtsteiner & Rocha (2002), ...)
- Signal & image processing, vision, OCR, ECG (Johnstone & Lu (2003))

Sparse PCA: Applications

What does sparsity mean here?

- *Financial time series analysis*: sparse factors often mean less assets in the portfolio, hence less fixed transaction costs
- *Multiscale data processing*: get sparse structure from motion data, ...
- *Gene expression data*: each variable is a particular gene, sparse factors highlight the action of a few genes, making interpretation easier
- *Image processing*: sparse factors involve only specific zones or objects in the image.

Related literature

Previous work:

- Cadima & Jolliffe (1995): the loadings with small absolute value are thresholded to zero.
- A non-convex method called SCoTLASS by Jolliffe & Uddin (2003). (Same setup here, numerical issues solved by relaxation)
- Zou, Hastie & Tibshirani (2004): a regression based technique called sparse PCA (S-PCA) (SPCA). Based on the fact that PCA can be written as a regression-type (non convex) optimization problem, using LASSO Tibshirani (1996) a l_1 norm penalty.

Performance:

- These methods are either very suboptimal or *nonconvex*
- Regression: works for *large scale* examples

A: rank one approximation

Problem definition:

- Here, we focus on the *first factor* x, computed as the solution of:

$$\min_{x \in \mathbf{R}} \|A - xx^T\|_F$$

  where $\|X\|_F$ is the Frobenius norm of X, i.e. $\|X\|_F = \sqrt{\mathbf{Tr}(X^2)}$

- In this case, we get an *exact* solution $\lambda^{\max}(A) x_1 x_1^T$ where $\lambda^{\max}(X)$ is the maximum eigenvalue and x_1 is the associated eigenvector.

Variational formulation

We can rewrite the previous problem as:

$$\begin{array}{ll} \text{max} & x^T A x \\ \text{subject to} & \|x\|_2 = 1. \end{array} \qquad (1)$$

Perron-Frobenius: this problem is *easy*, its solution is again $\lambda^{\max}(A)$ at x_1.

Here however, we want a little bit more. . .
We look for a *sparse* solution and solve instead:

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

where $\mathbf{Card}(x)$ denotes the cardinality (number of non-zero elements) of x.
This is non-convex and *numerically hard*.

Outline

- Introduction
- **Semidefinite relaxation**
- Large-scale problems
- Numerical results

Semidefinite relaxation

Start from:

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

let $X = xx^T$, and write everything in terms of the matrix X:

$$\begin{array}{ll} \text{max} & \mathbf{Tr}(AX) \\ \text{subject to} & \mathbf{Tr}(X) = 1 \\ & \mathbf{Card}(X) \leq k^2 \\ & X = xx^T. \end{array}$$

This is *strictly equivalent!*

Semidefinite relaxation

Why? If $X = xx^T$, then:

- in the objective: $x^T A x = \mathbf{Tr}(AX)$
- the constraint $\mathbf{Card}(x) \leq k$ becomes $\mathbf{Card}(X) \leq k^2$
- the constraint $\|x\|_2 = 1$ becomes $\mathbf{Tr}(X) = 1$.

We can go a little further and replace $X = xx^T$ by an equivalent $X \succeq 0,\;\; \mathbf{Rank}(X) = 1$, to get:

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

Again, this is the same problem!

Semidefinite relaxation

Numerically, this is still *hard*:

- The $\mathbf{Card}(X) \leq k^2$ is still non-convex
- So is the constraint $\mathbf{Rank}(X) = 1$

but, we have made *some progress*:

- The objective $\mathbf{Tr}(AX)$ is now *linear* in X
- The (non-convex) constraint $\|x\|_2 = 1$ became a *linear* constraint $\mathbf{Tr}(X) = 1$.

To solve this problem *efficiently*, we need to relax the two non-convex constraints above.

Semidefinite relaxation

Easy to do here. . .

If $u \in \mathbf{R}^p$, $\mathbf{Card}(u) = q$ implies $\|u\|_1 \leq \sqrt{q}\|u\|_2$. We transform the non-convex problem into a convex relaxation:

- Replace $\mathbf{Card}(X) \leq k^2$ by the weaker (*but convex*) $\mathbf{1}^T|X|\mathbf{1} \leq k$
- Simply drop the rank constraint

Our problem becomes now:

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

This is a convex program and can be solved *efficiently*.

Semidefinite programming

In fact, we get a semidefinite program in the variable $X \in \mathbf{S}^n$, which can be solved using *SEDUMI* by Sturm (1999) or *SDPT3* by Toh, Todd & Tutuncu (1996).

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

Complexity:

- Polynomial. . .
- Problem here: the program has $O(n^2)$ dense constraints on the matrix X.

In practice, hard to solve problems with $n > 15$ without additional work.

Singular Value Decomposition

Same technique works for Singular Value Decomposition instead of PCA.

- The variational formulation of *SVD* is here:

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

  in the variables $(u, v) \in \mathbf{R}^m \times \mathbf{R}^n$ where $k_1 \leq m$, $k_2 \leq n$ are fixed.

- This can be relaxed as the following *semidefinite program*:

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

  in the variable $X \in \mathbf{S}^{m+n}$ with blocks X_{ij} for $i, j = 1, 2$.

Outline

- Introduction
- Semidefinite relaxation
- **Large-scale problems**
- Numerical results

IP versus first-order methods

Interior Point methods for semidefinite/cone programs

- Produce a solution up to *machine precision*
- Compute a Newton step at each iteration: *costly*

In our case:

- We are not really interested in getting a solution up to machine precision
- The problems are *too big* to compute a Newton step. . .

Solution: use *first-order techniques*. . .

First-order methods

Basic model for the problem: *black-box* oracle producing

- the function value $f(x)$
- a subgradient $g(x) \in \partial f(x)$

f is here convex, non-smooth. Using only this info, we need $O(1/\varepsilon^2)$ steps to find an ε-optimal solution.

However, if the function is convex with a *Lipschitz-continuous gradient* with constant L then

- we need only $O\left(\sqrt{L/\varepsilon}\right)$ steps to get an ε-optimal solution.. . .

Smoothness brings a *massive* improvement in the complexity. . .

Sparse PCA?

In our case, we look at a penalized version of the relaxed sparse PCA problem:

$$\max_U \ \mathbf{Tr}(AU) - \mathbf{1}^T|U|\mathbf{1} \ : \ U \succeq 0, \ \ \mathbf{Tr}\, U = 1. \tag{5}$$

Difference?

- If we can solve the dual, these two formulations are equivalent.
- Otherwise: scale A. . .

Problem here, the function to minimize is not smooth! Can we hope to do better than the worst case complexity of $O(1/\varepsilon^2)$?

The answer is yes, exploit this particular *problem structure*. . .

Sparse PCA?

We can rewrite our problem as a *convex-concave* game:

$$\max_{\{U \succeq 0,\ \mathbf{Tr}\, U=1\}} \mathbf{Tr}(AU) - \mathbf{1}^T|U|\mathbf{1} = \min_{X \in \mathcal{Q}_1} \max_{U \in \mathcal{Q}_2} \langle X, U\rangle + \mathbf{Tr}(AU)$$

where

- $\mathcal{Q}_1 = \{X \in \mathcal{S}^n \ : \ |X_{ij}| \leq 1,\ \ 1 \leq i,j \leq n\}$
- $\mathcal{Q}_2 = \{U \in \mathcal{S}^n \ : \ \mathbf{Tr}\, U = 1\}$

Sparse PCA: complexity

Why a *convex-concave* game?

- Recent result by Nesterov (2003) shows that this specific structure can be exploited to significantly reduce the complexity compared to the black-box case
- All the algorithm steps can be worked out explicitly in this case

Algorithm in Nesterov (2003):

- reduces the complexity to $O\left(1/\varepsilon\right)$ instead of $O(1/\varepsilon^2)$!

Sparse PCA: large-scale algo.

We can formulate our problem using the notations in Nesterov (2003) (except for A becoming L here):

$$\max_{\{U \succeq 0,\ \mathbf{Tr}\, U=1\}} \mathbf{Tr}(AU) - \mathbf{1}^T|U|\mathbf{1} = \min_{X \in \mathcal{Q}_1} f(X)$$

where

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

Smooth minimization of non-smooth functions

What makes the algorithm in Nesterov (2003) work:

- First use the convex-concave game structure to regularize the function. (Inf-convolution with strictly convex function, à la Moreau-Yosida. See for example Lemaréchal & Sagastizábal (1997))

- Then use the optimal first-order minimization algorithm in Nesterov (1983) to minimize the smooth approximation.

The method works particularly well if:

- All the steps in the regularization can be performed in closed-form

- All the auxiliary minimization sub-problems can be solved in closed-form

It is the case here. . .

Regularization: prox functions

Procedure:

- First, we fix a regularization parameter μ
- Then, we define a *prox-function* for the set $\mathcal{Q}_2$:

$$d_2(U) = \mathbf{Tr}(U \log(U)) + \log(n), \quad U \in \mathcal{Q}_2$$

With this choice of d_2:

- the *center* of the set if then $X_0 = n^{-1} I_n$ with $d_2(X_0) = 0$
- the *convexity parameter* of d_2 on $\mathcal{Q}_2$ is bounded below by $\sigma_2 = 1/2$ (non-trivial, cf. Ben-Tal & Nemirovski (2004))

Regularization: prox functions

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

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

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

$$f_\mu(X) = \max_{U \in \mathcal{Q}_2} \langle X, U \rangle + \mathbf{Tr}(AU) - \mu d_2(U)$$

Because of our choice of prox-function:

- the function $f_\mu(X)$ *approximates* f with a maximum error of $\varepsilon/2$
- f_μ is *Lipschitz continuous* with constant:

$$L = \frac{1}{\mu \sigma_2}$$

Algorithm

Set the regularization parameter μ.

For $\mathbf{k \geq 0}$ do:

- Compute $f_\mu(X_k)$ and $\nabla f_\mu(X_k)$
- Find

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

- Find

$$Z_k = \arg\min_X \left\{ \frac{L}{\sigma_1} d_1(X) + \sum_{i=0}^{k} \frac{i+1}{2} \langle \nabla f_\mu(X_i), X - X_i \rangle \ : \ X \in \mathcal{Q}_1 \right\}$$

- Set $X_k = \frac{2}{k+3} Z_k + \frac{k+1}{k+3} Y_k$

Algorithm

Most expensive step is the first one, computing the value and gradient of f_μ:

- Compute $f_\mu(X)$ as

$$\max_{U \in \mathcal{Q}_2} \mathbf{Tr}(ZU) - \mu d_2(U), \quad \text{for } Z = A + X$$

- The gradient is the maximizer itself:

$$\nabla f_\mu(X) = \arg\max_{U \in \mathcal{Q}_2} \mathbf{Tr}(ZU) - \mu d_2(U)$$

The solution can be computed in *closed-form* as:

$$\mu \log \left(\sum_{i=1}^{n} \exp(\frac{\lambda_i(A+X)}{\mu}) \right) - \mu \log n$$

Algorithm

The second step can also be computed in *closed form*.

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

is equivalent to a *simple projection* problem:

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

Solution given by:

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

The third step is similar. . .

Convergence

- We can stop the algorithm when the gap

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

  where $U_k = u^*((A + X_k)/\mu)$ is our current estimate of the dual variable

- The above gap is necessarily non-negative, since both X_k and U_k are feasible for the primal and dual problem, respectively

Only check this criterion only periodically, for example every 100 iterations.

Complexity

- Max number of iterations is given by

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

with

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

- Since each iteration costs $O(n^3)$ flops, the worst-case flop count to get a ε-optimal solution is given by

$$O\left(\frac{n^4\sqrt{\log n}}{\epsilon}\right)$$

Outline

- Introduction
- Semidefinite relaxation
- Large-scale problems
- **Numerical results**

Cardinality versus k: model

Start with a sparse vector $v = (1,0,1,0,1,0,1,0,1,0)$. We then define the matrix A as:

$$A = U^T U + 15\ vv^T$$

here $U \in \mathbf{S}^{10}$ is a random matrix (uniform coefs in $[0,1]$).

We solve:

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

- Try $k = 1, \ldots, 10$
- For each k, sample a 100 matrices A
- Plot *average solution cardinality* (and standard dev. as error bars)

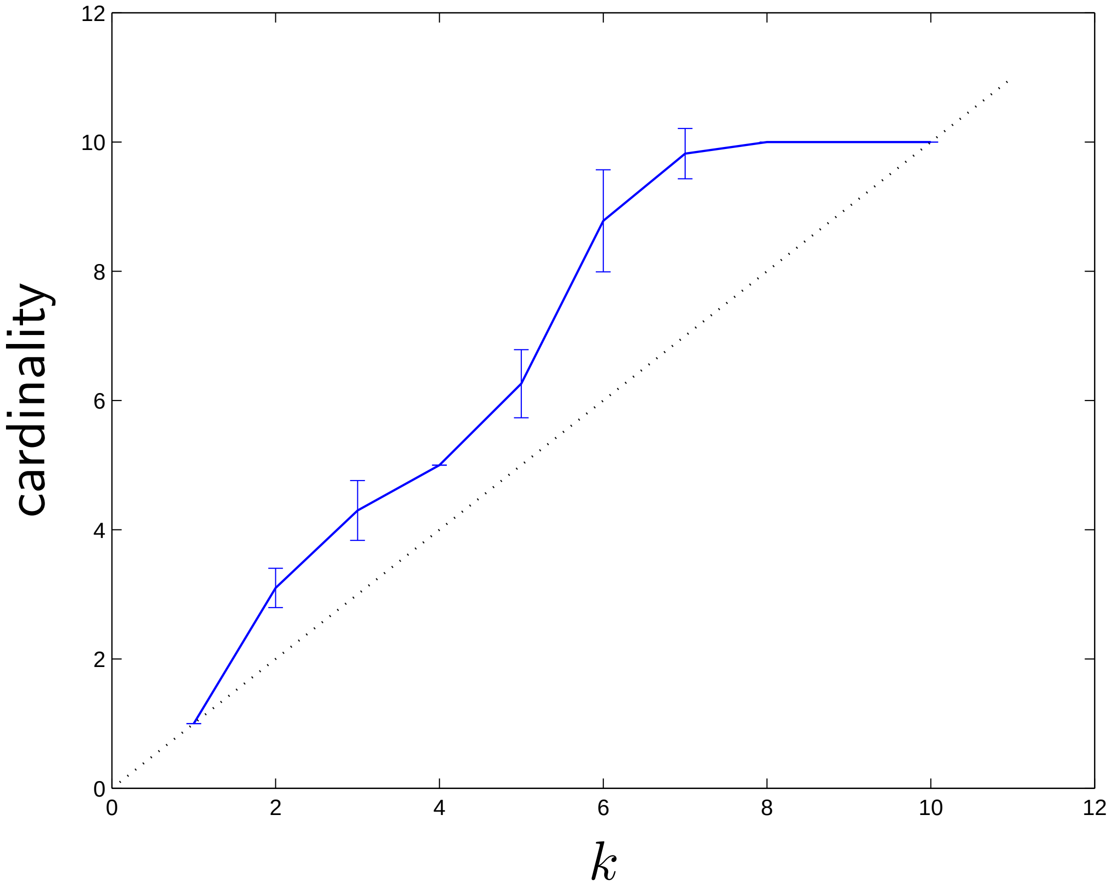


Figure 1: Cardinality versus k.

$(k+1)$ is a *very good predictor* of the cardinality. . .

Sparsity versus # iterations

Start with a sparse vector $v = (1, 0, 1, 0, 1, 0, 1, 0, 1, 0, \ldots, 0) \in \mathbf{R}^{20}$. We then define the matrix A as:

$$A = U^T U + 100\ vv^T$$

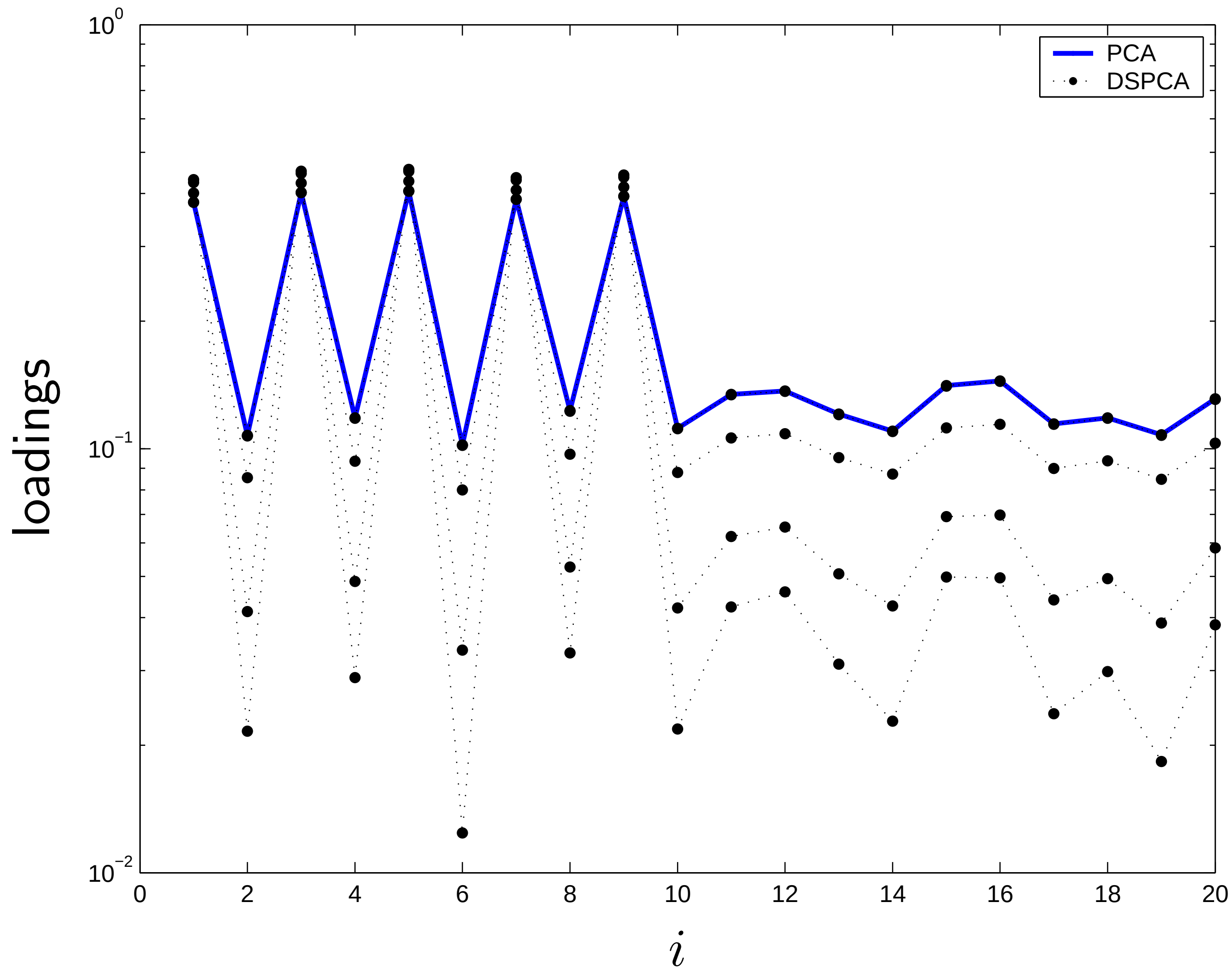
here $U \in \mathbf{S}^{20}$ is a random matrix (uniform coefs in $[0, 1]$).

We solve:

$$\begin{array}{ll} \text{max} & \mathbf{Tr}(AU) - \rho \mathbf{1}^T |U| \mathbf{1} \\ \text{s.t.} & \mathbf{Tr}\, U = 1 \\ & U \succeq 0 \end{array}$$

for $\rho = 5$.

Sparsity versus # iterations



Number of iterations: 10,000 to 100,000. Computing time: 12' to 110'.

References

Ben-Tal, A. & Nemirovski, A. (2004), 'Non-euclidean restricted memory level method for large-scale convex optimization', *MINERVA Working paper* .

Cadima, J. & Jolliffe, I. T. (1995), 'Loadings and correlations in the interpretation of principal components', *Journal of Applied Statistics* **22**, 203–214.

Chennubhotla, C. & Jepson, A. (2001), Sparse PCA: Extracting multi-scale structure from data, *in* 'International Conference on Computer Vision', IEEE, Vancouver, Canada, pp. 641–647.

Johnstone, I. & Lu, A. Y. (2003), Sparse principal components analysis, *in* 'FDA Workshop', Gainesville.

Jolliffe, I. T. & Uddin, M. (2003), 'A modified principal component technique based on the lasso', *Journal of Computational and Graphical Statistics* **12**, 531–547.

Lemaréchal, C. & Sagastizábal, C. (1997), 'Practical aspects of the Moreau-Yosida regularization: theoretical preliminaries', *SIAM Journal on Optimization* **7**(2), 367–385.

Nesterov, Y. (1983), 'A method of solving a convex programming problem with convergence rate $O(1/k^2)$', *Soviet Math. Dokl.* **27**(2), 372–376.

Nesterov, Y. (2003), 'Smooth minimization of nonsmooth functions', *CORE discussion paper 2003/12 (Accepted by Math. Prog.)* .

Rebonato, R. (1998), *Interest-Rate Options Models*, Financial Engineering, Wiley.

Sturm, J. F. (1999), 'Using sedumi 1.0x, a matlab toolbox for optimization over symmetric cones', *Optimization Methods and Software* **11**, 625–653.

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

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

Wall, M. E., Rechtsteiner, A. & Rocha, L. M. (2002), Singular value decomposition and principal component analysis, Technical Report ArXiv physics/0208101, Los Alamos National Laboratory.

Zou, H., Hastie, T. & Tibshirani, R. (2004), 'Sparse principal component analysis', *Technical report, statistics department, Stanford University* .