Sparse PCA with applications in finance

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

Princeton University, U.C. Berkeley & U.C. San Diego

Available *online* at www.princeton.edu/~aspremon

Introduction

Principal Component Analysis (*PCA*): 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.

Numerically, just an eigenvalue decomposition of the covariance matrix:

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

Portfolio Hedging

Hedging problem:

- Market is composed of N *assets* with price $S_{i,t}$ at time t
- Let C be the *covariance* matrix of the assets
- P_t is the value of a *portfolio* of assets with coefficients u_i:

$$P_t = \sum_{i=1}^{N} u_i S_{i,t}$$

- The market *factors* and corresponding variances are given by:

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

Portfolio Hedging

- We can hedge some of the risk using the k most important *market factors*:

$$P_t = \sum_{i=1}^{k} (u^T x_i) F_{i,t} + \varepsilon_t, \quad \text{with } F_{i,t} = x_i^T S_t$$

- Usually $k = 3$. On interest rate markets the first three factors are *level*, *spread* and *convexity*.

- Problem: the factors x_i usually assign a *nonzero* weight to *all* assets S_i

- This means large *fixed transaction costs* when hedging. . .

Sparse PCA: Applications

Can we get *sparse* factors x_i instead?

- *Portfolio hedging*: sparse factors mean less assets in the portfolio, hence less transaction costs.

- *Side effects*: minimize proportional transaction costs, robustness interpretation.

- *Other applications*: image processing, gene expression data analysis, multi-scale data processing.

Variational formulation

We can rewrite the previous problem as:

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

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:

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

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

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, Trendafilov & Uddin (2003). (Same problem formulation)
- Zou, Hastie & Tibshirani (2004): a regression based technique called SPCA. Based on a representation of PCA as a regression problem. Sparsity is obtained using the LASSO Tibshirani (1996) a l_1 norm penalty.

Performance:

- These methods are either very suboptimal (thresholding) or lead to *nonconvex* optimization problems (SPCA).
- Regression: works for very *large scale* examples.

Semidefinite relaxation

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 *a strictly equivalent* problem.

Semidefinite relaxation

From

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

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

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

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

If $u \in \mathbf{R}^p$, $\mathbf{Card}(u) = q$ implies $\|u\|_1 \leq \sqrt{q}\|u\|_2$. Hence, we can find 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:

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

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

Semidefinite programming

More specifically, 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).

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

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

Solution here: use first order algorithm developed by Nesterov (2005).

Robustness

Dual

We look at the penalized problem:

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

which can be written:

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

or also:

$$\min_{\{|X_{ij}| \leq \rho\}} \quad \lambda^{\max}(A + X)$$

This dual has a *very natural interpretation*. . .

Dual: robust PCA

The dual problem is:

$$\min_{\{|X_{ij}|\leq\rho\}} \quad \lambda^{\max}(A+X)$$

- Worst-case *robust* maximum eigenvalue problem
- Componentwise noise with magnitude ρ on the coefficients of the covariance matrix A

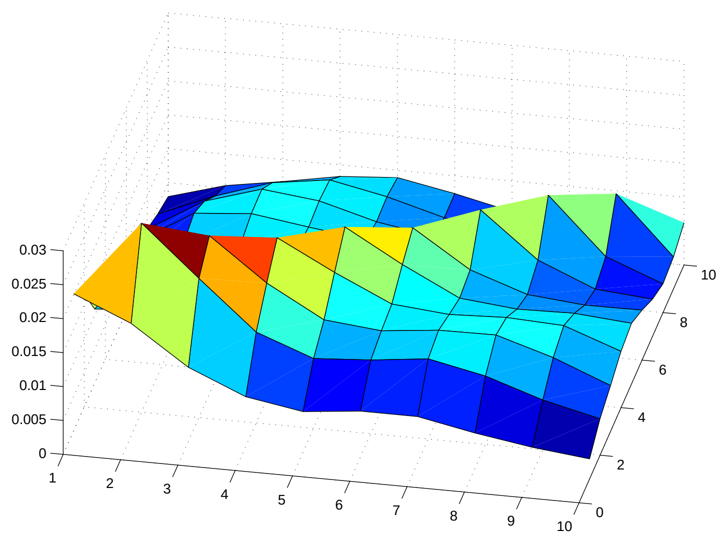
Asking for *sparsity* in the primal means solving a *robust* maximum eigenvalue problem with uniform noise on the coefficients.

Numerical results

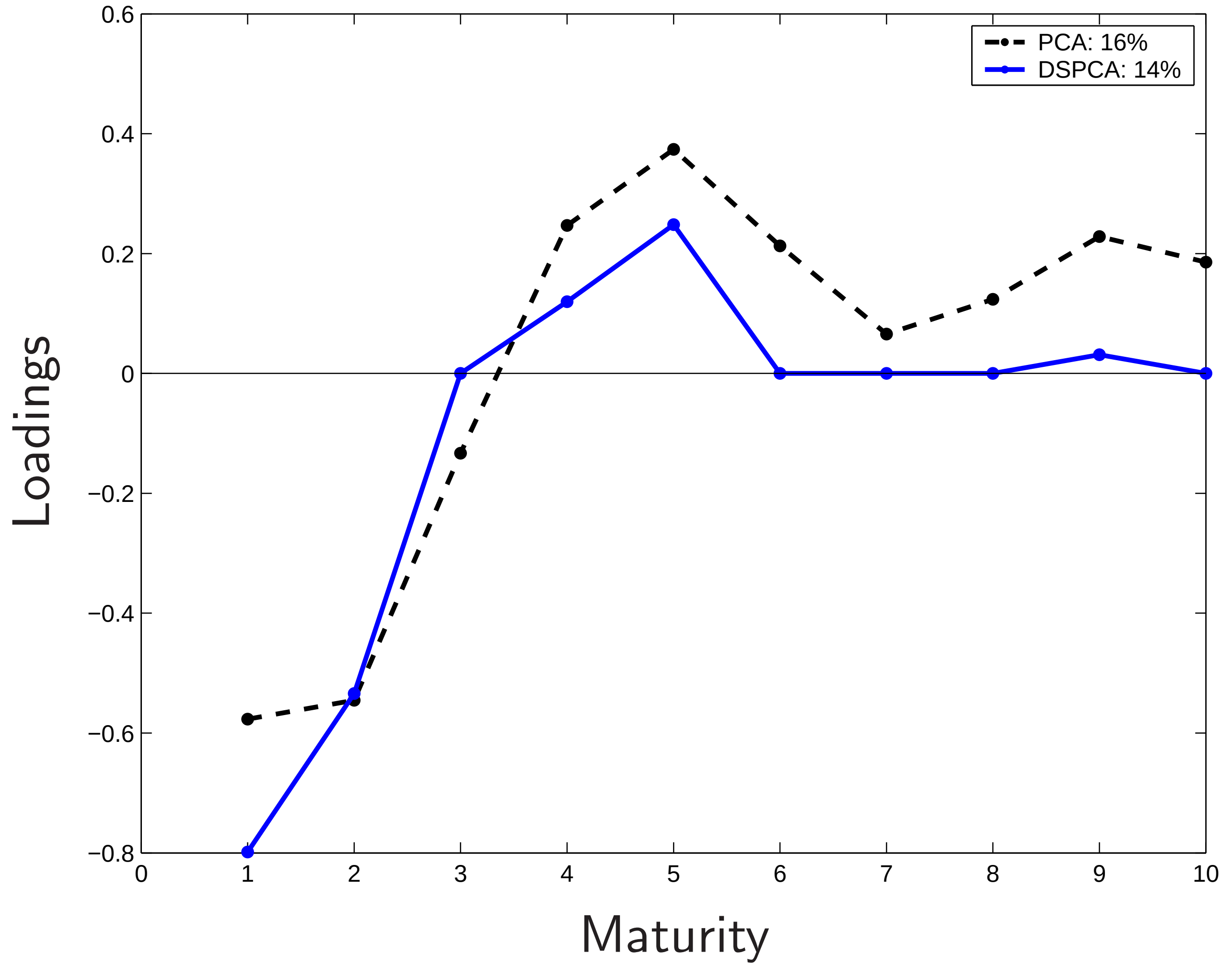
Sparse factors. . .

Example:

- Use a covariance matrix from forward rates with maturity 1Y to 10Y
- Compute first factor normally (average of rates)
- Use the relaxation to get a sparse *second factor*



Second Factor



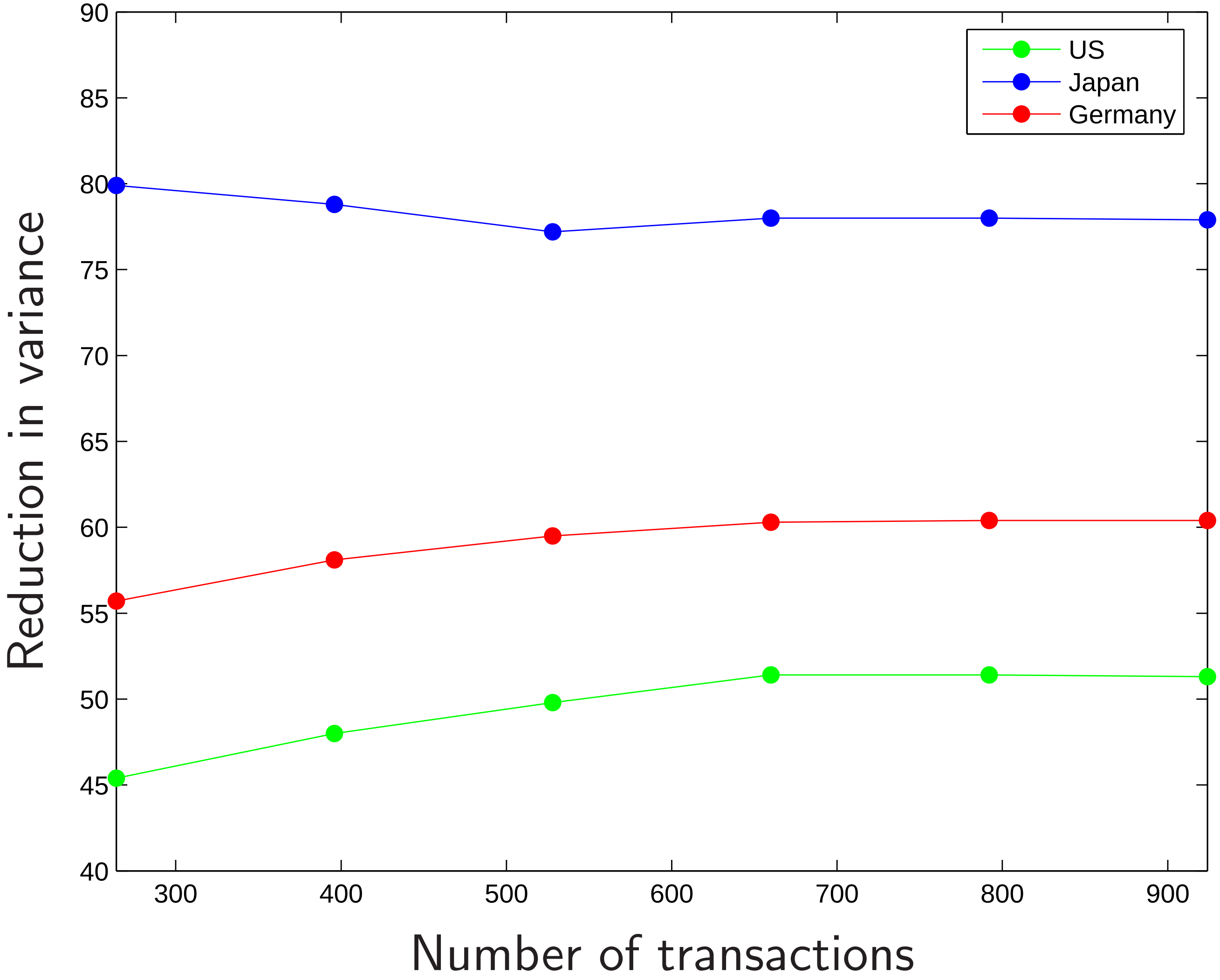
The second factor is much sparser than in the PCA case (5 nonzero components instead of 10), explained variance goes from 16% to 14%. . .

Portfolio hedging

- Pick a random portfolio of forward rates in JPY, USD and EUR
- Hedge it and compute the residual variance over a three months horizon
- Hedge only using the first factor
- Record the percentage reduction in variance for various levels of sparsity

(Thanks to Aslheigh Kreider for research assistance)

Portfolio hedging



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:

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

where $U \in \mathbf{S}^{10}$ 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)

Cardinality versus k

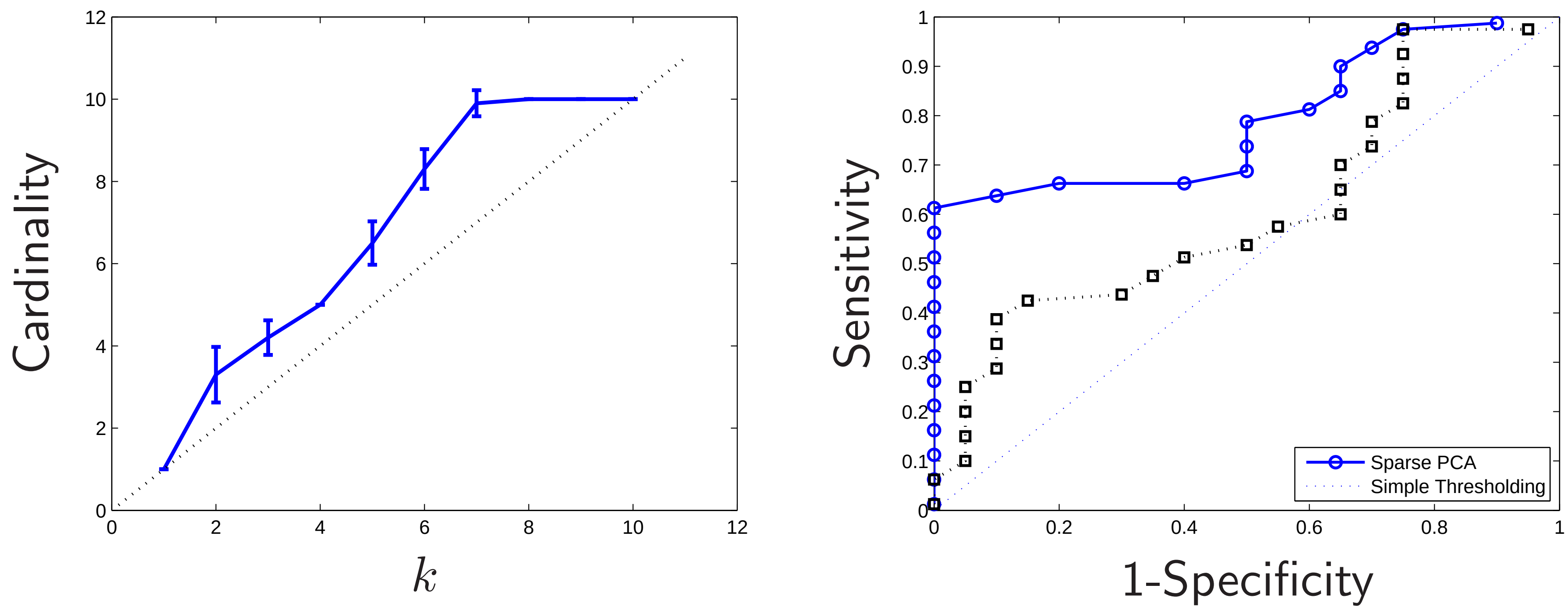


Figure 1: Cardinality versus k. ROC curves

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:

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

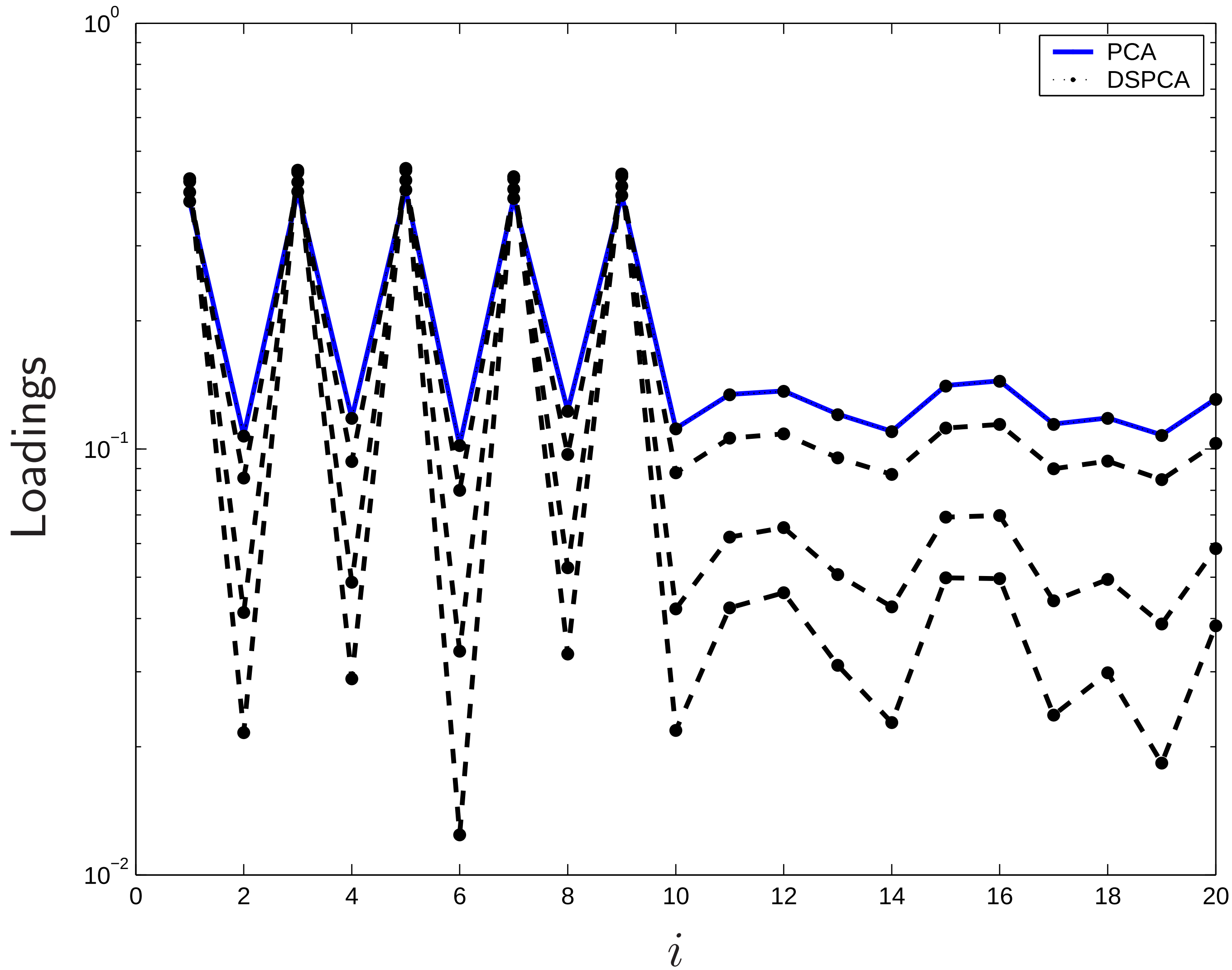
here $U \in \mathbf{S}^{20}$ is a random matrix (uniform coefs in $[0,1]$).

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".

Conclusion

- *Semidefinite relaxation* for sparse PCA
- *Robustness* & *sparsity* at the same time (cf. dual)
- Can solve large-scale problems with first-order method by Nesterov (2005)
- (Approximately) optimal factors when fixed transaction costs are present

Slides and software available *online* at `www.princeton.edu/~aspremon`

References

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

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

Nesterov, Y. (2005), 'Smooth minimization of nonsmooth functions', *Mathematical Programming, Series A* **103**, 127–152.

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.

Zou, H., Hastie, T. & Tibshirani, R. (2004), 'Sparse principal component analysis', *To appear in JCGS* .