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
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.
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?
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
Problem definition:

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

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

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

- In this case, we get an *exact* solution $\lambda_{\text{max}}(A)x_1x_1^T$ where $\lambda_{\text{max}}(X)$ is the maximum eigenvalue and $x_1$ is the associated eigenvector.
Variational formulation

We can rewrite the previous problem as:

$$\begin{align*}
\text{max} & \quad x^T Ax \\
\text{subject to} & \quad \|x\|_2 = 1.
\end{align*}$$ (1)

Perron-Frobenius: this problem is \textit{easy}, its solution is again $\lambda_{\text{max}}(A)$ at $x_1$.

Here however, we want a little bit more. . .

We look for a \textit{sparse} solution and solve instead:

$$\begin{align*}
\text{max} & \quad x^T Ax \\
\text{subject to} & \quad \|x\|_2 = 1 \\
& \quad \text{Card}(x) \leq k,
\end{align*}$$ (2)

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

Outline

• Introduction

• **Semidefinite relaxation**

• Large-scale problems

• Numerical results
Start from:

\[
\begin{align*}
\max & \quad x^T Ax \\
\text{subject to} & \quad \|x\|_2 = 1 \\
& \quad \text{Card}(x) \leq k,
\end{align*}
\]

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

\[
\begin{align*}
\max & \quad \text{Tr}(AX) \\
\text{subject to} & \quad \text{Tr}(X) = 1 \\
& \quad \text{Card}(X) \leq k^2 \\
& \quad X = xx^T.
\end{align*}
\]

This is strictly equivalent!
Semidefinite relaxation

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

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

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

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

(3)

Again, this is the same problem!
Semidefinite relaxation

Numerically, this is still *hard*:

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

but, we have made *some progress*:

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

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

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

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

Our problem becomes now:

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

This is a convex program and can be solved efficiently.
Semidefinite programming

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

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

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 \textit{SVD} is here:

\[
\begin{align*}
\min & \quad \|A - uv^T\|_F \\
\text{subject to} & \quad \text{Card}(u) \leq k_1 \\
& \quad \text{Card}(v) \leq k_2,
\end{align*}
\]

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

- This can be relaxed as the following \textit{semidefinite program}:

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

in the variable \(X \in \mathbb{S}^{m+n}\) 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 $\varepsilon$-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 $\varepsilon$-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:

$$\max_U \text{Tr}(AU) - 1^T|U|1 : U \succeq 0, \text{ Tr } U = 1.$$  \hspace{1cm} (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 \textit{problem structure}. . .
We can rewrite our problem as a *convex-concave* game:

$$\max_{\{U \succeq 0, \tr U = 1\}} \tr(AU) - 1^T|U|1 = \min_{X \in Q_1} \max_{U \in Q_2} \langle X, U \rangle + \tr(AU)$$

where

- $Q_1 = \{X \in S^n : |X_{ij}| \leq 1, 1 \leq i, j \leq n\}$
- $Q_2 = \{U \in S^n : \tr U = 1\}$
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 $O\left(\frac{1}{\varepsilon}\right)$ instead of $O\left(\frac{1}{\varepsilon^2}\right)!$
We can formulate our problem using the notations in Nesterov (2003) (except for $A$ becoming $L$ here):

$$\max_{\{U \succeq 0, \ Tr \ U = 1\}} Tr(AU) - 1^T|U|1 = \min_{X \in Q_1} f(X)$$

where

- $Q_1 = \{X \in S^n : |X_{ij}| \leq 1, \ 1 \leq i, j \leq n\}$
- $f(X) = \lambda_{\max}(A + X) = \max_{U \in Q_2} \langle BX, U \rangle - \hat{\phi}(U)$
- $Q_2 = \{U \in S^n : \ Tr \ U = 1\}, \ B = I_{n^2}, \ \hat{\phi}(U) = - Tr(AU)$
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 $\mu$

• Then, we define a prox-function for the set $Q_2$:

$$d_2(U) = \text{Tr}(U \log(U)) + \log(n), \quad U \in Q_2$$

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 $Q_2$ is bounded below by $\sigma_2 = 1/2$
Regularization: prox functions

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

\[ \min_{X \in \mathcal{Q}_1} f_\mu(X), \]

where \( f_\mu \) is the penalized function involving the prox-function \( d_2 \):

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

Because of our choice of prox-function:

- the function \( f_\mu(X) \) *approximates* \( f \) with a maximum error of \( \varepsilon/2 \)

- \( f_\mu \) is *Lipschitz continuous* with constant:

\[ L = \frac{1}{\mu \sigma^2} \]
Set the regularization parameter $\mu$.

**For $k \geq 0$ do:**

- Compute $f_\mu(X_k)$ and $\nabla f_\mu(X_k)$
- Find
  $$Y_k = T_{Q_1}(X_k) = \arg \min_{Y \in Q_1} \langle \nabla f_\mu(X), Y - X \rangle + \frac{1}{2} L \|X - Y\|_F^2$$
- Find
  $$Z_k = \arg \min_X \left\{ \frac{L}{\sigma_1} d_1(X) + \sum_{i=0}^{k} \frac{i + 1}{2} \langle \nabla f_\mu(X_i), X - X_i \rangle : X \in Q_1 \right\}$$
- 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 $f_\mu$:

- Compute $f_\mu(X)$ as

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

- The gradient is the maximizer itself:

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

The solution can be computed in \textit{closed-form} as:

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

The second step can also be computed in \textit{closed form}.

\[ Y_k = T_{Q_1}(X_k) = \arg \min_{Y \in Q_1} \langle \nabla f_{\mu}(X), Y - X \rangle + \frac{1}{2} L \|X - Y\|_F^2 \]

is equivalent to a \textit{simple projection} problem:

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

Solution given by:

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

The third step is similar. . .
Convergence

- We can stop the algorithm when the gap

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

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

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

- Since each iteration costs \( O(n^3) \) flops, the worst-case flop count to get a \( \epsilon \)-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 $U \in S^{10}$ is a random matrix (uniform coefs in $[0, 1]$).

We solve:

$$\max \quad \text{Tr}(AX)$$

subject to

$$\text{Tr}(X) = 1$$
$$1^T |X| 1 \leq k$$
$$X \succeq 0,$$

- 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 $v = (1, 0, 1, 0, 1, 0, 1, 0, \ldots, 0) \in \mathbb{R}^{20}$. We then define the matrix $A$ as:

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

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

We solve:

$$\max \quad \text{Tr}(AU) - \rho \mathbf{1}^T |U| \mathbf{1}$$

s.t.  
$$\text{Tr} \, U = 1$$
$$U \succeq 0$$

for $\rho = 5$. 

Sparsity versus # iterations

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

References


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


