WIRTSCHAFTS
UNIVERSITÄT
WIEN VIENNA
UNIVERSITY OF
ECONOMICS
AND BUSINESS

Sparse Principal Component Analysis

Formulations And Algorithms

Outline

Principal Component Analysis - Problem I

- Task: projection of a or covariance matrix X (or centered data matrix X) from $\boldsymbol{R}^p$ into $\boldsymbol{R}^q, \quad q \leq p$
- PCA is a sequence of projections onto a linear manifold in $\boldsymbol{R}^q$
- The projections to $\boldsymbol{R}^q$ should explain the maximum amount of data variance unaccounted for
- The projections should be orthogonal

There are two equivalent representations of that problem

- Minimize the reconstruction error for a centered data matrix (least squares problem) $\rightarrow$ singluar value decomposition
- Maximize the variance explained over the linear combination for a given covariance matrix (eigenvalue problem) $\rightarrow$ eigenstructure

PCA - Problem II

We will focus on the maximum variance formulation (cf. Sharma, 1996): PCA looks to reconstruct the covariance matrix of X, $E(X^TX)$ by linear combinations of the original p variables, i.e. finding v_q for $v_q^T E(X^TX) v_q$. This leads to

PCA - Maximum Variance

$$\max v_q^T (X^TX) v_q \text{ subject to } v_q^T v_q = 1; v_{q-1}^T v_q = 0. \tag{1}$$

with v_q denoting the q-th eigen vector of X^TX with loadings for each of the p variables ($1 \leq q \leq p$). The other formulation for which the reconstruction error is minimized is

$$min_{\lambda, V_q} \sum_{i=1}^{n} \|x_i - V_q \lambda_i\|^2$$

V_q is a $p \times q$ matrix with orthogonal unit vectors in columns.

PCA - Solution

Solving (1) with Lagrange multipliers λ leads to

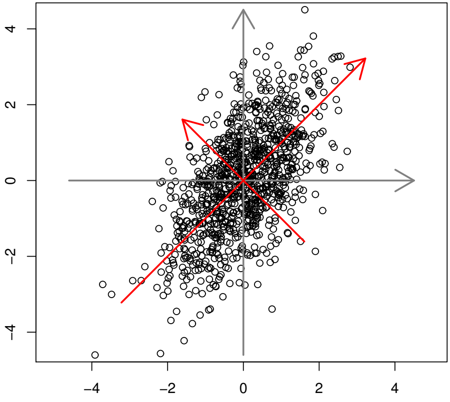
$$(X^T X - \lambda I)v_q = 0 \tag{2}$$

hence an eigenvalue problem. There are p (non-trivial) eigenvalues $\lambda_q, (q = 1, \ldots, p)$ and for each we get the corresponding eigenvectors by

$$v_q^T (X^T X) v_q = \lambda_q \tag{3}$$

v_q is called the q-th principal component and λ_q is the variance explained by the linear combination.

PCA - Representation



Sparse PCA - Motivation

The PCA solution usually leads to all p loadings $v_{.j}, j = 1, \ldots, p$ being non zero.

- Interpretation can be more difficult
- Problematic if p is very large

The idea of sparse PCA is now to force a number of $v_{.j}$ to be zero, hence the eigenvector is sparse.

Sparse PCA - Motivation II

For example, consider this application:

- Asset allocation: We want to get principal components of the Eurostoxx 50, i.e. linearly reconstruct the covariance matrix. Additionally we want to derive portfolio allocation weights but minimize transaction costs (hence not invest in all 50 stocks, but, say, 5).

We are sure one can find many applications in economics, text mining, business analytics and so on.

Formulations Of Sparse PCA Problem

The sparse PCA problem can again be formulated as variance maximization subject to a side condition (cardinality problem) or via recontruction error (leads to a elastic net problem).

sPCA - Maximum Variance

$$\max v_q^T (X^T X) v_q$$
$$\text{subject to} \quad v_q^T v_q = 1$$
$$\|v_q\|_0 \leq k$$

or

$$\min_{\theta, v_q} \sum_{i=1}^{n} \|x_i - \theta v_q^T x_i\|_2^2 + \delta \|v_q\|_2^2 + \delta_1 \|v_q\|_1 \text{ subject to } \theta^T \theta = 1$$

Optimisation Of Sparse PCA Problem

Unfortunately, optimisation of the aforementioned problems are not trivial, it is a combinatorial problem.

- Both are non convex optimization problems. The latter is convex for parameter subsets fixed.
- Computation is complex and cumbersome, especially for the first.
- The second one needs large scale examples.

ScoTLASS, Elasticnet and Others

Algorithms that we will not consider further:

- Sorting simply sorts the diagonal of the covariance matrix and ranks the variables by variance
- Thresholding computes the leading eigenvectors and form sparse vector by thresholding all coefficients below a certain level
- SCotLASS (Joliffe et al., 2003) solves the maximum variance problem. It is nonconvex.
- Elastic Net (Zhou et al., 2006) solves the regression problem. It is implemented in R in the package elasticnet.

Optimality By Semidefinite Relaxation

d'Aspremont et al. (2008) formulate (4) as a semidefinite relaxation problem and derive greedy algorithms to solve it, as well as conditions for global optimality of a solution. Their approach delivers

- Greedy algorithms for computing a full set of solutions ($k = 1, \ldots, q$)
- Tractable sufficient conditions for v_q to be a global optimum of (4)
- Numerical cost: $O(p^3)$

Yet Another Formulation

Actually, they use

$$\phi(\rho) = \max_{\|v_1\| \leq 1} v_1^T (X^T X) v_1 - \rho \|v_1\|_0 \tag{4}$$

which is directly related to (4) (ρ denotes the sparsity parameter). Mostly this means that an optimal solution for the latter is globally optimal for (4). Note the rank one approximation. Reformulating (4) gives

$$\begin{aligned} \phi(\rho) = \max_{\|z\|=1} \sum_{i=1}^{p} ((x_i^T z)^2 - \rho)_+ \quad &= \quad \max \sum_{i=1}^{p} (x_j^T z z^T x_j - \rho)_+ \\ &s.t. \quad Tr(zz^T) = 1, Rank(zz^T) = 1, zz^T \succeq 0 \end{aligned}$$

with the $\cdot_+$ operator denoting $\max\{0, \cdot\}$.

Semidefinite Relaxation

This can be written as a semidefinite program in the variables $Z = zz^T$ and P_i

$$\psi(\rho) = \max \sum_{i=1}^{p} Tr(P_i B_i) \tag{5}$$

$$s.t. \quad Tr(Z) = 1, Z \succeq 0, Z \succeq P_i \succeq 0 \tag{6}$$

with $B_i = x_i x_i^T - \rho I$. It always holds that $\psi(\rho) \geq \phi(\rho)$ and when program has rank one, equality holds $\psi(\rho) \geq \phi(\rho)$. This can be used to derive global optimality conditions for the original problem.

Greedy Search Algorithm

Moghaddam et al. (2006) proposed a greedy algorithm to solve (4): Input a positive semidefinite, symmetric matrix Σ

1. Decreasingly sort diagonal of $\Sigma = X^T X$ and permute elements accordingly. Compute Cholesky decomposition of Σ
2. Initialisation: $I_1 = \{1\}, z_1 = x_1/\|x_1\|$
3. Compute $i_k = argmax_{i \notin I_k} \lambda_{max}(\sum_{j \in I_k \cup \{i\}} x_j x_j^T)$
4. Set $I_{k+1} = I_k \cup \{i_k\}$ and compute z_{k+1} as the leading eigenvector of $\sum_{j \in I_{k+1}} x_j x_j^T$
5. Set $k = k + 1$. If $k < p$ go to step 3.

Outputs sparsity patterns I_k. Costs: $O(p^4)$.
At each step $v_k = argmax_{\{v_{I_k^c}, \|v\|=1\}} v^T \Sigma v - \rho k$, the solution to (4) given I_k.

Approximate Greedy Search Algorithm

d'Aspremont et al. (2008) proposed an approximate greedy algorithm to solve (4): Input a positive semidefinite, symmetric matrix Σ

1. Decreasingly sort diagonal of $\Sigma = X^T X$ and permute elements accordingly. Compute Cholesky decomposition of σ.
2. Initialisation: $I_1 = \{1\}, z_1 = x_1/\|x_1\|$
3. Compute $i_k = argmax_{i \notin I_k} (z_k^T x_i)^2$
4. Set $I_{k+1} = I_k \cup \{i_k\}$ and compute z_{k+1} as the leading eigenvector of $\sum_{j \in I_{k+1}} x_j x_j^T$
5. Set $k = k + 1$. If $k < p$ go to step 3.

Outputs sparsity patterns I_k. Costs: $O(p^3)$.
At each step $v_k = argmax_{\{v_{I_k^c}, \|v\|=1\}} v^T \Sigma v - \rho k$, the solution to (4) given I_k.

Sparse PCA in R

Currently, there is only one implementation: `spca` in elasticnet. Hence we started implementing these algorithms

- `spca_greedy()`: The full blown greedy search
- `spca_approx()`: The approximate greedy search
- `check_optimality()`: Check optimality of a sparsity pattern (see later)

Later we will see five additional algorithms. Also, there are some other matlab and python functions available that we might port to R. Package?

Illustration - Eurostoxx 15/50

Let us assume the market are $p = 15$ assets from Eurostoxx 50 with price $P_{i,t}$ at time t. We have a covariance matrix S of assets. P_t is the value of a portfolio of assets with coefficients f_i, $P_t = \sum_{i=1}^{p} f_i P_{i,t}$. We built a matrix of asset time series $(2007-01-08$ to $2011-23-05)$ via

```
> load("finDatClean.rda")
> dim(finDat)

[1] 959  15

> matplot(finDat, type = "l")
```

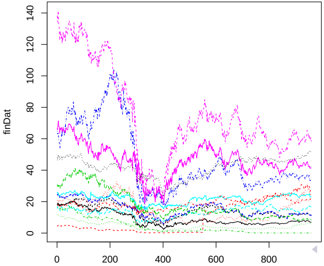


Illustration - Eurostoxx 15/50

The so-called market factors are given by

$$S = \sum_{i=1}^{p} \lambda_i v_i v_i^T \tag{7}$$

Apparently, one can hedge some of the risk using the q most important factors

$$P_t = \sum_{i=1}^{q} (f^T v_i) F_{i,t} + e_t \text{ with } F_{i,t} = v_i^T P_{i,t} \tag{8}$$

Usually, $q = 3$ is chosen (called level, spread and convexity). The factors v_i usually assign a nonzero weight to all assets P_i, which might lead to large transaction costs.

With classic PCA:

```
> sc.finDat <- scale(finDat, scale = FALSE)
> res.pca <- prcomp(sc.finDat)
> res.pca$rotation[, 1:3]

                 PC1           PC2          PC3
AEG.Close   0.114962728 -0.0188007519  0.015491510
AI.Close   -0.078663443  0.7044274925 -0.331108266
ALU.Close   0.052552556  0.0251466159 -0.002019747
MT.Close    0.526412767 -0.2880767757 -0.646760447
G.Close     0.028738769  0.1660812118 -0.065879896
CS.Close    0.219421295  0.3041230051  0.283754596
STD.Close   0.100528689  0.0224251968  0.001224187
BAS.Close   0.126508648 -0.0395491507 -0.407513694
BAYN.Close  0.108791799 -0.0839864905  0.071668968
BBVA.Close  0.119082885 -0.0004836227  0.069992034
CA.Close    0.051488262  0.1109290267 -0.073840617
DB.Close    0.735583013  0.2278055151  0.380195301
DTE.Close   0.067747294  0.3782475023 -0.207584048
ENI.Close  -0.005149654  0.1880536092 -0.071303175
NOK.Close   0.226440656 -0.2190369003  0.112636246
```

With sparse PCA

```
> source("code/pathSPCA.R")
> res.spca <- spca_approx(dat = sc.finDat, k = 5)
> colnames(sc.finDat)[res.spca$index]

[1] "DB.Close"  "MT.Close"  "NOK.Close" "CS.Close"  "BAS.Close"

> finSub <- sc.finDat[, res.spca$index]
> res.pcaSub <- prcomp(finSub)
> res.pcaSub$rotation[, 1:3]

                PC1        PC2        PC3
DB.Close  0.7613117  0.4512434 -0.1155939
MT.Close  0.5451746 -0.7008459  0.2008054
NOK.Close 0.2329092 -0.0514374 -0.7647086
CS.Close  0.2273997  0.4242716  0.5542596
BAS.Close 0.1312702 -0.3500603  0.2330930
```

Illustration - Eurostoxx

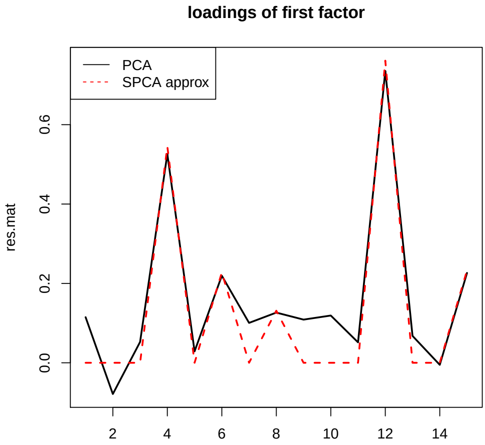
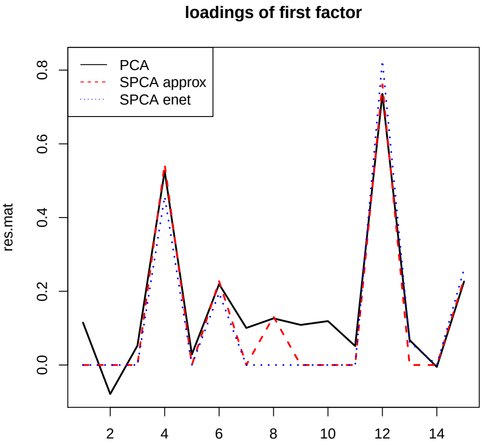


Illustration - Eurostoxx 15/50

To compare with elastic net

```
> library(elasticnet)
> res.enet <- spca(x = sc.finDat, K = 5, type = "predictor", sparse = "varnum",
+     para = c(5, 5, 5, 5, 5), trace = FALSE)
> res.enet$loadings[, 1:3]

                  PC1         PC2         PC3
AEG.Close  0.00000000  0.00000000  0.00000000
AI.Close   0.00000000  0.87819572  0.00000000
ALU.Close  0.00000000  0.00000000  0.00000000
MT.Close   0.45681778  0.00000000 -0.87704443
G.Close    0.00000000  0.11037742  0.00000000
CS.Close   0.19522480  0.00000000  0.09407484
STD.Close  0.00000000  0.00000000  0.00000000
BAS.Close  0.00000000  0.00000000  0.00000000
BAYN.Close 0.00000000  0.00000000  0.15739780
BBVA.Close 0.00000000  0.00000000  0.00000000
CA.Close   0.00000000  0.00000000  0.00000000
DB.Close   0.82569075  0.00000000  0.41061560
DTE.Close  0.06273822  0.44151222  0.00000000
ENI.Close  0.00000000  0.07890564  0.00000000
NOK.Close  0.25981436 -0.12421738  0.16900817
```


loadings of first factor
PCA
SPCA approx
SPCA enet
res.mat
0.0
0.2
0.4
0.6
0.8
2
4
6
8
10
12
14

Optimality Condition

One can check the global optimality of a sparsity pattern I with the sufficient condition from d'Aspremont et. al. (2008, Theorem 6): Let v be the largest eigenvector of $\sum_{i \in I} x_i x_i^T$. If there is a $\rho* \geq 0$ such that

$$max_{i \in I^c}(x_i^T v)^2 < \rho* < min_{i \in I^c}(v_i^T x)^2 \text{ and } \lambda_{max}(\sum_i Y_i) \leq \sum_{i \in I}((x_i^T v)^2 - \rho*), \tag{9}$$

and Y_i are the dual variables and it holds that if $i \in I$

$$Y_i = \frac{B_i v v^T B_i}{v^T B_i v}, \tag{10}$$

then I is globally optimal for (4) with $\rho = \rho*$ and the optimal solution v can be obtained by solving the unconditional PCA problem for the submatrix.

Optimality Condition - R

For the greedy algorithm

```
> check_optimality(Sigma = res.spca$Sigma, subset = res.spca$res)

$optimal
[1] TRUE

$rho
[1] 21800
```

For the elasticnet algorithm

```
> check_optimality(Sigma = res.spca$Sigma, subset = c(1, 2, 5,
+     3, 12))

$optimal
[1] FALSE

$rho
[1] NA
```

Only the first subset is optimal.

Generalized Power Method

See you at 17.6.

References

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

I. Joliffe, N. Trendafilov, and M. Uddin. A modified principal component technique based on the lasso. *Journal of Computational and Graphical Statistics*, 12:531–547, 2003.

B. Moghaddam, Y. Weiss, and S. Avidan. Spectral bounds for sparse pca: Exact and greedy algorithms. *Advances in Neural Information Preocessing Systems*, 18, 2006.

S. Sharma. *Applied Multivariate Techniques*. Wiley, 1996.

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

Thank you for your attention

Thomas Rusch, Norbert Walchhofer

WU Wirtschaftsuniversität Wien
Augasse 2–6, A-1090 Wien