Sparse Principal Component Analysis
Formulations And Algorithms
Outline

1 Background
   - What Is Principal Component Analysis (PCA)?
   - What Is Sparse Principal Component Analysis (sPCA)?

2 The Sparse PCA Problem
   - Formulations of sPCA
   - Optimisation in sPCA

3 Algorithmic solutions for sPCA
   - ScotLASS and elasticnet
   - Semidefinite Relaxation And Optimal Solution
   - Generalized Power Method
Task: projection of a or covariance matrix $X$ (or centered data matrix $X$) from $\mathbb{R}^p$ into $\mathbb{R}^q$, $q \leq p$

PCA is a sequence of projections onto a linear manifold in $\mathbb{R}^q$

The projections to $\mathbb{R}^q$ 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) $\rightarrow$ singular 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 $v_q^TE(X^TX)v_q$. This leads to

**PCA - Maximum Variance**

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

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.
Solving (1) with Lagrange multipliers $\lambda$ leads to

$$(X^TX - \lambda I)v_q = 0$$  \hspace{1cm} (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

$v_q^T(X^TX)v_q = \lambda_q$  \hspace{1cm} (3)$$

$v_q$ is called the $q$-th principal component and $\lambda_q$ is the variance explained by the linear combination.
PCA - Representation
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.
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.
The sparse PCA problem can again be formulated as variance maximization subject to a side condition (cardinality problem) or via reconstruction error (leads to an elastic net problem).

**sPCA - Maximum Variance**

\[
\max_{v_q} v_q^T (X^T X) v_q \\
\text{subject to} \quad v_q^T v_q = 1 \\
\|v_q\|_0 \leq k
\]

or

\[
\min_{\theta, v_q} \sum_{i=1}^{n} \|x_i - \theta v_q^T x_i\|_2^2 + \delta \|v_q\|_2^2 + \delta_1 \|v_q\|_1 \quad \text{subject to} \quad \theta^T \theta = 1
\]
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.
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 forms a 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.
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: \(O(p^3)\)
Actually, they use

\[ \phi(\rho) = \max_{\|v_1\| \leq 1} v_1^T (X^T X) v_1 - \rho \| v_1 \|_0 \]  \hspace{1cm} (4) 

which is directly related to (4) (\( \rho \) 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

\[ \phi(\rho) = \max_{\|z\| = 1} \sum_{i=1}^{p} ((x_i^T z)^2 - \rho)_+ = \max \sum_{i=1}^{p} (x_j^T zz^T x_j - \rho)_+ \]

s.t. \( Tr(zz^T) = 1, \text{Rank}(zz^T) = 1, zz^T \succeq 0 \)

with the \( \cdot_+ \) operator denoting \( \max\{0, \cdot\} \).
This can be written as a semidefinite program in the variables $Z = zz^T$ and $P_i$

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

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

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 $\psi(\rho) = \phi(\rho)$. This can be used to derive global optimality conditions for the original problem.
Moghaddam et al. (2006) proposed a greedy algorithm to solve (4): Input a positive semidefinite, symmetric matrix $\Sigma$.

1. Decreasingly sort diagonal of $\Sigma = X^T X$ and permute elements accordingly. Compute Cholesky decomposition of $\Sigma$.

2. Initialisation: $I_1 = \{1\}$, $z_1 = x_1 / \|x_1\|$.

3. Compute $i_k = \arg\max_{i \not\in I_k} \lambda_{\max} \left( \sum_{j \in I_k \cup \{i\}} x_j x_j^T \right)$.

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 $v_k = \arg\max_{\{v_{i_k}, \|v\| = 1\}} v^T \Sigma v - \rho k$, 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 $\Sigma$

1. Decreasingly sort diagonal of $\Sigma = X^T X$ and permute elements accordingly. Compute Cholesky decomposition of $\sigma$.

2. Initialisation: $I_1 = \{1\}$, $z_1 = x_1 / ||x_1||$

3. Compute $i_k = \text{argmax}_{i \notin I_k} (z_k^T x_i)^2$

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 $v_k = \text{argmax}_{\{v \in I_k, ||v||=1\}} v^T \Sigma v - \rho k$, the solution to (4) given $I_k$. 
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?
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

\[
\text{> load("finDatClean.rda")}
\text{> dim(finDat)}
\]

\[ [1] \; 959 \; 15 \]

\[
\text{> matplot(finDat, type = "l")}
\]
The so-called market factors are given by

\[ S = \sum_{i=1}^{p} \lambda_i \mathbf{v}_i \mathbf{v}_i^T \]  

(7)

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

\[ P_t = \sum_{i=1}^{q} (f^T \mathbf{v}_i) F_{i,t} + e_t \text{ with } F_{i,t} = \mathbf{v}_i^T P_{i,t} \]  

(8)

Usually, \( q = 3 \) is chosen (called level, spread and convexity). The factors \( \mathbf{v}_i \) usually assign a nonzero weight to all assets \( P_i \), which might lead to large transaction costs.
With classic PCA:

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

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

```r
> 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]

<table>
<thead>
<tr>
<th></th>
<th>PC1</th>
<th>PC2</th>
<th>PC3</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB.Close</td>
<td>0.7613117</td>
<td>0.4512434</td>
<td>-0.1155939</td>
</tr>
<tr>
<td>MT.Close</td>
<td>0.5451746</td>
<td>-0.7008459</td>
<td>0.2008054</td>
</tr>
<tr>
<td>NOK.Close</td>
<td>0.2329092</td>
<td>-0.0514374</td>
<td>-0.7647086</td>
</tr>
<tr>
<td>CS.Close</td>
<td>0.2273997</td>
<td>0.4242716</td>
<td>0.5542596</td>
</tr>
<tr>
<td>BAS.Close</td>
<td>0.1312702</td>
<td>-0.3500603</td>
<td>0.2330930</td>
</tr>
</tbody>
</table>
```
Illustration - Eurostoxx 15/50

loadings of first factor

PCA
SPCA approx

SLIDE 22  RS Optimisation
To compare with elastic net

```r
> 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 0.0000000 0.0000000 0.0000000
AI.Close  0.0000000 0.8781957 0.0000000
ALU.Close 0.0000000 0.0000000 0.0000000
MT.Close  0.4568178 0.0000000 -0.8770444
G.Close   0.0000000 0.1103774 0.0000000
CS.Close  0.1952248 0.0000000 0.0940748
STD.Close 0.0000000 0.0000000 0.0000000
BAS.Close 0.0000000 0.0000000 0.0000000
BAYN.Close 0.0000000 0.0000000 0.1573978
BBVA.Close 0.0000000 0.0000000 0.0000000
CA.Close  0.0000000 0.0000000 0.0000000
DB.Close  0.8256907 0.0000000 0.4106156
DTE.Close 0.0627382 0.4415122 0.0000000
ENI.Close 0.0000000 0.0789056 0.0000000
NOK.Close 0.2598143 -0.1242173 0.1690082
```
Illustration - Eurostoxx 15/50

loadings of first factor

- PCA
- SPCA approx
- SPCA enet

res.mat

SLIDE 24  RS Optimisation
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

$$\max_{i \in I^c} (x_i^T v)^2 < \rho^* < \min_{i \in I^c} (v_i^T x)^2$$

and

$$\lambda_{\max}(\sum_i Y_i) \leq \sum_{i \in I} ((x_i^T v)^2 - \rho^*),$$

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

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

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

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

$optimal
[1] TRUE

$rho
[1] 21800
```

For the elasticnet algorithm

```r
> 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.


Thank you for your attention

Thomas Rusch, Norbert Walchhofer
WU Wirtschaftsuniversität Wien
Augasse 2–6, A-1090 Wien