sparseWeightBasedPCA Package

Niek C. de Schipper n.c.deschipper@uvt.nl

2020-06-19

Contents

`sparseWeightBasedPCA`: A package for Regularized weight based Simultaneous Component Analysis (SCA) and Principal Component Analysis (PCA)

Theoretical background

Principal Component Analysis

Principal component analysis (PCA) is a widely used analysis technique for data reduction. It can give crucial insights in the underlying structure of the data when used as a latent variable model.

Given a data matrix $\mathbf{X}$ that contains the scores for $i = 1...I$ observations on $j = 1...J$ variables; we follow the convention to present the J variable scores of observation i in row i and thus $\mathbf{X}$ has size $I \times J$. PCA decomposes the data into Q components as follows,

$$\begin{aligned} &\mathbf{X} = \mathbf{XWP}^T + \mathbf{E} \\ &\text{subject to } \mathbf{P}^T\mathbf{P} = \mathbf{I}, \end{aligned} \tag{1}$$

where $\mathbf{W}$ is a $J \times Q$ component weight matrix, $\mathbf{P}$ is a $J \times Q$ loading matrix and $\mathbf{E}$ is a $I \times J$ residual matrix. The component weight matrix $\mathbf{W}$ will be the focus of this package, note that $\mathbf{T} = \mathbf{XW}$ represent the component scores.

The advantage of inspecting the component weights instead of the loadings is that you can directly derive meaning to $\mathbf{T}$, this because you see precisely in what way items in $\mathbf{X}$ are weighted together by $\mathbf{W}$.

Simultaneous Component Analysis

The decomposition in (1) can be extended to the case of multi-block data by taking $\mathbf{X}_c = [\mathbf{X}_1 \dots \mathbf{X}_K]$; this is concatenating the K data blocks composed of different sets of variables of size J_k for the same units of observation. The decomposition of $\mathbf{X}_c$ has the same block structured decomposition as in (1) with $\mathbf{W}_c = [\mathbf{W}_1^T \dots \mathbf{W}_K^T]^T$ and $\mathbf{P}_c = [\mathbf{P}_1^T \dots \mathbf{P}_K^T]^T$. This multi-block formulation of PCA is known as simultaneous component analysis (SCA):

$$\begin{aligned} &[\mathbf{X}_1 \dots \mathbf{X}_K] = [\mathbf{X}_1 \dots \mathbf{X}_K][\mathbf{W}_1^T \dots \mathbf{W}_K^T]^T[\mathbf{P}_1^T \dots \mathbf{P}_K^T] + \mathbf{E} \\ &\text{subject to } [\mathbf{P}_1^T \dots \mathbf{P}_K^T][\mathbf{P}_1^T \dots \mathbf{P}_K^T]^T = \mathbf{I} \end{aligned} \tag{2}$$

When analyzing multi-block data with SCA identifying meaningful relations between data blocks is of prime interest. In order to gain insight in what multiple data blocks relate to each other, we can search for blockwise structures in the component weights that tell us whether a component is uniquely determined by variables from one single data block (distinctive component), or whether it is a component that is determined by variables from multiple data blocks (common component). In other words, a distinctive component is a linear combination of variables of a particular data block only, whereas a common component is a linear combination of variables of multiple data blocks. An example of common and distinctive components in the situation with two

data blocks is given below. The first two components are distinctive components, the third component is a common component,

$$\mathbf{T} = \begin{bmatrix}\mathbf{X}_1 & \mathbf{X}_2\end{bmatrix}\begin{bmatrix}\mathbf{W}_1 \\ \mathbf{W}_2\end{bmatrix} = \begin{bmatrix}\mathbf{X}_1 & \mathbf{X}_2\end{bmatrix}\begin{bmatrix} 0 & w_{1_1 2} & w_{1_1 3} \\ 0 & w_{2_1 2} & w_{2_1 3} \\ 0 & w_{3_1 2} & w_{3_1 3} \\ w_{1_2 1} & 0 & w_{1_2 3} \\ w_{2_2 1} & 0 & w_{2_2 3} \\ w_{3_2 1} & 0 & w_{2_2 3}\end{bmatrix}.$$

The `sparseWeightBasedPCA` package will provide functions that perform PCA and SCA on the component weights. It will also provide function for selection of the hyper parameters of the model. We will now describe the core models of this package that will be estimated by the following functions.

1. `scads` Regularized SCA with sparse component weights using constraints
2. `mmsca` Regularized SCA with sparse component weights using the group LASSO
3. `ccpca` PCA with sparse component weights using cardinality constraints

WRITE HERE WHAT OTHER PACKAGES DO

Models of the `sparseWeightBasedPCA` package

Regularized SCA with sparse component weights using constraints

Here we present an approach of performing regularized SCA, with ridge and LASSO regularization and block wise constraints on $\mathbf{W}_c$ by solving,

$$\begin{aligned} L(\mathbf{W}_c, \mathbf{P}_c) =& \|\mathbf{X}_c - \mathbf{X}_c\mathbf{W}_c\mathbf{P}_c^T\|_2^2 + \lambda_L\|\mathbf{W}_c\|_1 + \lambda_R\|\mathbf{W}_c\|_2^2 \\ & \text{subject to } \mathbf{P}_c\mathbf{P}_c^T = \mathbf{I}, \text{ and } \lambda_L, \lambda_R \geq 0 \text{ and zero block constraints on } \mathbf{W}_c \end{aligned} \tag{3}$$

In order to get a minimum for () we alternate between the estimation of $\mathbf{W}_c$ and $\mathbf{P}_c$. Given $\mathbf{W}_c$ we can estimate $\mathbf{P}_c$ by using procrustes rotation. Given $\mathbf{P}_c$ we find estimates for $\mathbf{W}_c$ by using a coordinate descent algorithm that works by soft-thresholding the weights. For the specifics we refer the reader to (REF: NIEK KATRIJN). This iterative procedure stops when an optimum has been found (.i.e the loss function value is not decreasing anymore beyond pre-specified tolerance level). The optimization problem in (4) is non-convex and meaning there are local minima. In order to deal with that multiple random starts can be used with different initializations of $\mathbf{W}$, the start leading to the lowest evaluation of (4) is retained. Typically starting the algorithm with the solution of PCA (e.g. the first Q right singular vectors of $\mathbf{X}$) will lead to smallest optimum.

The main advantage of analyzing multi-block data by using this procedure is that it is fast, and scalable to large data sets thanks to the coordinate descent implementation. The inclusion of the blockwise constraints on $\mathbf{W}$ make sure common and distinctive components are found and the LASSO and ridge regularizers are optional and facilitate extra sparsity within the component weights. A disadvantage of the method is that the common and distinctive structures for $\mathbf{W}$ need to be selected. This has to be done using model selection procedures and can be computationally demanding.

This procedure has been implemented in the `scads` function. This function will be discussed in detail in the next section and examples will be given outlining the analysis including model selection.

Regularized SCA with sparse component weights using the group LASSO

Here we present a very flexible approach of performing regularized SCA using, ridge, LASSO, group LASSO and elitist LASSO regularization by solving:

$$\begin{aligned} L(\mathbf{W}_c, \mathbf{P}_c) =& \|\mathbf{X}_c - \mathbf{X}_c\mathbf{W}_c\mathbf{P}_c^T\|_2^2 + \lambda_L\|\mathbf{W}_c\|_1 + \lambda_R\|\mathbf{W}_c\|_2^2 \\ & + \sum_{q,k}(\lambda_G\sqrt{J_k}\|\mathbf{w}_q^{(k)}\|_2 + \lambda_E\|\mathbf{w}_q^{(k)}\|_{1,2}) \\ & \text{subject to } \mathbf{P}_c\mathbf{P}_c^T = \mathbf{I} \text{ and } \lambda_L, \lambda_R, \lambda_G, \lambda_E \geq 0 \end{aligned} \tag{4}$$

where $\mathbf{W}_c = [(\mathbf{W}^{(1)})^T \dots (\mathbf{W}^{(K)})^T]^T$, and $\mathbf{w}_q^{(k)}$ denotes the qth column from the submatrix $\mathbf{W}^{(k)}$. In order to get a minimum for (4) we alternate between the estimation of $\mathbf{W}_c$ and $\mathbf{P}_c$. Given $\mathbf{W}_c$ we can estimate $\mathbf{P}_c$ by using procrustes rotation. Given $\mathbf{P}_c$ we can find estimates for $\mathbf{W}_c$ by using the majorization minimization (MM) algorithm. For the specifics we refer the reader to (REF: NIEK KATRIJN). This iterative procedure stops when an optimum has been found (.i.e the loss function value is not decreasing anymore beyond pre-specified tolerance level). The optimization problem in (4) is non-convex and meaning there are local minima. In order to deal with that multiple random starts can be used with different initializations of $\mathbf{W}$, the start leading to the lowest evaluation of (4) is retained. Typically starting the algorithm with the solution of PCA (e.g. the first Q right singular vectors of $\mathbf{X}$) will lead to smallest optimum.

The main advantage of solving (4) is that it can automatically look for common and distinctive components by taking advantage of the properties of the group lasso. Because the group LASSO is specified on the colored segments (see below), it will either include these segments or put them zero, uncovering common and distinctive components. This is especially useful if the number of blocks and components is large, and an exhaustive approach of identifying common and distinctive is too computationally intensive.

$$\mathbf{T} = \begin{bmatrix}\mathbf{X}_1 & \mathbf{X}_2\end{bmatrix}\begin{bmatrix}\mathbf{W}_1 \\ \mathbf{W}_2\end{bmatrix} = \begin{bmatrix}\mathbf{X}_1 & \mathbf{X}_2\end{bmatrix}\begin{bmatrix} w_{1_1 1} & w_{1_1 2} & w_{1_1 3} \\ w_{2_1 1} & w_{2_1 2} & w_{2_1 3} \\ w_{3_1 1} & w_{3_1 2} & w_{3_1 3} \\ w_{1_2 1} & w_{1_2 2} & w_{1_2 3} \\ w_{2_2 1} & w_{2_2 2} & w_{2_2 3} \\ w_{3_2 1} & w_{3_2 2} & w_{2_2 3} \end{bmatrix}.$$

The inclusion the LASSO and ridge regularization are optional and facilitate extra sparsity within the colored segments. The elitist LASSO has a very special use case, the elitist LASSO will include all colored segments and will put weights within each segment to zero. The elitist lasso can be used to force components to be common. It is not advised to use the group LASSO and the elitist LASSO together as they have opposing goals. A disadvantage of using this procedure is that is potentially slow, this because its implemented using a MM-algorithm which tend to be slow in convergence.

This procedure has been implemented in the `mmsca` function. This function will be discussed in detail in the next section and examples will be given outlining the analysis including model selection.

PCA with sparse component weights using cardinality constraints

Here we present an approach of solving PCA by applying cardinality constraints to the component weights by solving:

$$\begin{aligned} &L(\mathbf{W}, \mathbf{P}) = \|\mathbf{X} - \mathbf{X}\mathbf{W}\mathbf{P}^T\|_2^2 \\ &\text{subject to } \mathbf{W} \text{ including } K \text{ zeros.} \end{aligned} \tag{5}$$

In order to get a minimum for (5) we need to alternate between the estimation of $\mathbf{W}$ and $\mathbf{P}$. Given $\mathbf{W}$ we can estimate $\mathbf{P}$ by using procruste rotation (REF: Ten_Berge_2005, Zou_2006), $\mathbf{P} = \mathbf{U}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are the left and right singular vectors of $\mathbf{X}^T\mathbf{X}\mathbf{W}$. Given $\mathbf{P}$ we can find estimates for $\mathbf{W}$ given the cardinality constraints using the cardinality constraint regression algorithm for detail see (REF: NIEK KATRIJN). The optimization problem in (5) is non-convex and meaning there are local minima. In order to deal with that multiple random starts can be used with different initializations of $\mathbf{W}$, the start leading to the lowest evaluation of (5) is retained. Typically starting the algorithm with the solution of PCA (e.g. the first Q right singular vectors of $\mathbf{X}$) will lead to smallest optimum.

The main advantage of solving (5) is that this model tries to directly tackle the problem of finding the underlying subset of weights, in contrast to the usage of a penalty that shrinks the weights and also induces sparsity such as the LASSO. This approach can lead to better discovery of the underlying weights compared to LASSO (REF: NIEK KATRIJN). Another advantage is that you directly impose cardinality constraints on $\mathbf{W}$. This gives the user total control over the amount of sparsity. This can be desirable if there is already an idea about the level of sparsity in the final model. A disadvantage of using this procedure is that is potentially slow, this because the CCREG algorithm uses a MM-algorithm which tend to be slow in convergence. Another potential downside could be the absence of regularizers, they tend to shrink the variance of the estimators leading to more efficiency. In noisy situations other procedures might outperform this procedure.

This model has been implemented in the `ccpca` function. This function will be discussed in detail in the next section.

Overview of available functions in the `sparseWeightBasedPCA` package

The `sparseWeightBasedPCA` provides functions for the aforementioned models, also model selection procedures are provided in order to tune the hyper parameters. The main functions: `scads, mmsca` and `ccpca` are implemented in `C++` using the packages Rcpp and RcppArmadillo. The model selection procedures are implemented in `R`. The functions of the package will be discussed after which examples will be given.

`scads`

- `X`: A data matrix of class 'matrix'
- `ncomp`: The number of components to estimate (an integer)
- `ridge`: A numeric value containing the ridge parameter for the component weight matrix W
- `lasso`: A vector containing a ridge parameter for each column of W separately, to set the same lasso penalty for the component weights W, specify: lasso = rep(value, ncomp)
- `constraints`: A matrix of the same dimensions as the component weights matrix W (ncol(X) x ncomp). A zero entry corresponds in constraints corresponds to an element in the same location in W that needs to be constraint to zero. A non-zero entry corresponds to an element in the same location in W that needs to be estimated.

- `itr`: The maximum number of iterations (an integer)
- `Wstart`: A matrix of ncomp columns and nrow(X) rows with starting values for the component weight matrix W, if Wstart only contains zeros, a warm start is used: the first ncomp right singular vectors of X
- `tol`: The convergence is determined by comparing the loss function value after each iteration, if the difference is smaller than tol, the analysis is converged. The default value is 10e-8.
- `nStarts`: The number of random starts the analysis should perform. The first start will be performed with the values given by Wstart. The consecutive starts will be
- `Wstart`plus a matrix with random uniform values times the current start number (the first start has index zero).
- `printLoss`: A boolean: TRUE will print the lossfunction value each 10th iteration.

Value:

```
A list containing:
‘W’ A matrix containing the component weights
‘P’ A matrix containing the loadings
‘loss’ A numeric variable ccontaining the minumn loss function
value of all the nStarts starts
‘converged’ A boolean containing ‘TRUE’ if converged ‘FALSE’ if
not converged.
```