Package: mmpca (via r-universe)

October 4, 2024

Title Integrative Analysis of Several Related Data Matrices

Version 2.0.3

Description A generalization of principal component analysis for integrative analysis. The method finds principal components that describe single matrices or that are common to several matrices. The solutions are sparse. Rank of solutions is automatically selected using cross validation. The method is described in Kallus et al. (2019) <arXiv:1911.04927>.

Depends R (>= 3.3.0)

Imports digest (>= 0.6.0), Rcpp (>= 1.0.8)

LinkingTo Rcpp, RcppEigen, RcppGSL

SystemRequirements C++14

Biarch true

NeedsCompilation yes

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.2

URL https://github.com/cyianor/mmpca

BugReports https://github.com/cyianor/mmpca/issues

Author Jonatan Kallus [aut], Felix Held [ctb, cre]

Maintainer Felix Held <felix.held@gmail.com>

Repository CRAN

Date/Publication 2022-11-15 08:50:02 UTC

Contents

mmpca *Multiview principal component analysis*

Description

Analyzes several related matrices of data.

Usage

```
mmpca(
  x,
  inds,
  k,
  lambda = NULL,
  trace = 0,
  max_iter = 20000,
  init_theta = NULL,
  cachepath = NULL,
  enable_rank_selection = TRUE,
  enable_sparsity = TRUE,
  enable_variable_selection = FALSE,
  parallel = TRUE
)
```

Arguments

x	List of matrices to analyze
inds	Matrix containing view indices. The matrix should have two columns and the same number of rows as the length of x. The first (second) column contains the view index of the rows (columns) of the corresponding matrix.
k	Integer giving the maximum rank of the analysis, i.e. the maximum number of principal components for each view.
lambda	Vector or matrix of lambda values. The length (or width if it is a matrix) depends on the number of active penalties (2, 3 or 4). If it is a matrix, try different lambda values (one try for each row). Default: a matrix where each column is the sequence exp(seq(-6, 0))).
trace	Integer selecting the amount of log messages. 0 (default): no output, 3: all output.
max_iter	Maximum number of iterations
init_theta	NULL, functions or numeric. NULL (default) use initial values based on ordinary SVD. If init_theta is a list of three functions (CMF, matrix_to_triplets and getCMFopts from package CMF) use the supplied functions to find initial values with collaborative matrix factorization (CMF). If init_theta is a numeric vector it is used as initial value.

`cachepath` Character vector with path to directory to store intermediate results. If NULL (default) intermediate results are not stored. For caching to work it is required that the random number generation seed is constant between calls to mmpca, so `set.seed` needs to be called before mmpca.

`enable_rank_selection` Boolean deciding if the second penalty that imposes a low rank model should be enabled.

`enable_sparsity` Boolean deciding if the third penalty that imposes sparsity in V should be enabled.

`enable_variable_selection` Boolean deciding if the fourth penalty that increases the tendency for sparsity structure of different V columns to be similar. Defaults to FALSE meaning this penalty is not used.

`parallel` Boolean deciding if computations should be run on multiple cores simultaneously.

Value

A list with components

`initial` initial values used in optimization

`cmf` solution found with CMF (if init_theta == c(CMF, matrix_to_triplets, getCMFopts))

`training` solutions for different values of lambda

`solution` solution for optimal lambda value

Author(s)

Jonatan Kallus, <kallus@chalmers.se>

Examples

```
# Create model with three views, two data matrices of low-rank 3
max_rank <- 3
v <- list(
  qr.Q(qr(matrix(rnorm(10 * max_rank), 10, max_rank))),
  qr.Q(qr(matrix(rnorm(11 * max_rank), 11, max_rank))),
  qr.Q(qr(matrix(rnorm(12 * max_rank), 12, max_rank)))
)
d <- matrix(
  c(1, 1, 1, 1, 1, 0, 1, 0, 1),
  nrow = max_rank, ncol = 3
)
x <- list(
  v[[1]] %*% diag(d[, 1] * d[, 2]) %*% t(v[[2]]),
  v[[1]] %*% diag(d[, 1] * d[, 3]) %*% t(v[[3]])
)
inds <- matrix(c(1, 1, 2, 3), 2, 2)
```

```
result <- mmpca::mmpca(
  x, inds, max_rank, parallel = FALSE,
  lambda = c(1e-3, 1e-5), enable_sparsity = FALSE,
  trace = 3
)
# Investigate the solution
result$solution$D
```

Index