Package: nipals (via r-universe)

September 14, 2024

Title Principal Components Analysis using NIPALS or Weighted EMPCA, with Gram-Schmidt Orthogonalization

Version 0.9

Date 2021-09-14

Description Principal Components Analysis of a matrix using Non-linear Iterative Partial Least Squares or weighted Expectation Maximization PCA with Gram-Schmidt orthogonalization of the scores and loadings. Optimized for speed. See Andrecut (2009) <doi:10.1089/cmb.2008.0221>.

License MIT + file LICENSE

URL https://kwstat.github.io/nipals/

BugReports https://github.com/kwstat/nipals/issues

Depends R (>= 3.4.0)

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Repository https://kwstat.r-universe.dev

RemoteUrl https://github.com/kwstat/nipals

RemoteRef HEAD

RemoteSha 765bef02a739cdb18f6ec493342fcc248b1bd5d2

Contents

avg_angular_distance *Average angular distance between two rotation matrices*

Description

For matrices A and B, calculate the angle between the column vectors of A and the corresponding column vectors of B. Then average the angles.

Usage

```
avg_angular_distance(A, B)
```

Arguments

A	Matrix
B	Matrix

Details

The results of the singular value decomposition X=USV' are unique, but only up to a change of sign for columns of U, which indicates that the axis is flipped.

Value

A single floating point number, in radians.

Author(s)

Kevin Wright

References

None

Examples

```
# Example from https://math.stackexchange.com/questions/2113634/
rot1 <- matrix(c(-0.956395958, -0.292073218, 0.000084963,
                 0.292073230, -0.956395931, 0.000227268,
                 0.000014880, 0.000242173, 0.999999971),
               ncol=3, byrow=TRUE)
rot2 <- matrix(c(-0.956227882, -0.292623029, -0.000021887,
                 0.292623030, -0.956227882, -0.000024473,
                 -0.000013768, -0.000029806, 0.999999999),
               ncol=3, byrow=TRUE)
avg_angular_distance(rot1, rot2) # .0004950387
```

empca *Principal component analysis by weighted EMPCA, expectation maximization principal component-analysis*

Description

Used for finding principal components of a numeric matrix. Missing values in the matrix are allowed. Weights for each element of the matrix are allowed. Principal Components are extracted one a time. The algorithm computes x = TP', where T is the 'scores' matrix and P is the 'loadings' matrix.

Usage

```
empca(
  x,
  w,
  ncomp = min(nrow(x), ncol(x)),
  center = TRUE,
  scale = TRUE,
  maxiter = 100,
  tol = 1e-06,
  seed = NULL,
  fitted = FALSE,
  gramschmidt = TRUE,
  verbose = FALSE
)
```

Arguments

x	Numerical matrix for which to find principal components. Missing values are allowed.
w	Numerical matrix of weights.
ncomp	Maximum number of principal components to extract from x.
center	If TRUE, subtract the mean from each column of x before PCA.
scale	if TRUE, divide the standard deviation from each column of x before PCA.
maxiter	Maximum number of EM iterations for each principal component.
tol	Default 1e-6 tolerance for testing convergence of the EM iterations for each principal component.
seed	Random seed to use when initializing the random rotation matrix.
fitted	Default FALSE. If TRUE, return the fitted (reconstructed) value of x.
gramschmidt	Default TRUE. If TRUE, perform Gram-Schmidt orthogonalization at each iteration.
verbose	Default FALSE. Use TRUE or 1 to show some diagnostics.

Value

A list with components `eig`, `scores`, `loadings`, `fitted`, `ncomp`, `R2`, `iter`, `center`, `scale`.

Author(s)

Kevin Wright

References

Stephen Bailey (2012). Principal Component Analysis with Noisy and/or Missing Data. Publications of the Astronomical Society of the Pacific. http://doi.org/10.1086/668105

Examples

```
B <- matrix(c(50, 67, 90, 98, 120,
              55, 71, 93, 102, 129,
              65, 76, 95, 105, 134,
              50, 80, 102, 130, 138,
              60, 82, 97, 135, 151,
              65, 89, 106, 137, 153,
              75, 95, 117, 133, 155), ncol=5, byrow=TRUE)
rownames(B) <- c("G1","G2","G3","G4","G5","G6","G7")
colnames(B) <- c("E1","E2","E3","E4","E5")
dim(B) # 7 x 5
p1 <- empca(B)
dim(p1$scores) # 7 x 5
dim(p1$loadings) # 5 x 5

B2 = B
B2[1,1] = B2[2,2] = NA
p2 = empca(B2, fitted=TRUE)
```

---

nipals *Principal component analysis by NIPALS, non-linear iterative partial least squares*

Description

Used for finding principal components of a numeric matrix. Missing values in the matrix are allowed. Principal Components are extracted one a time. The algorithm computes x = TP', where T is the 'scores' matrix and P is the 'loadings' matrix.

Usage

```
nipals(
  x,
  ncomp = min(nrow(x), ncol(x)),
```

```
  center = TRUE,
  scale = TRUE,
  maxiter = 500,
  tol = 1e-06,
  startcol = 0,
  fitted = FALSE,
  force.na = FALSE,
  gramschmidt = TRUE,
  verbose = FALSE
)
```

Arguments

x	Numerical matrix for which to find principal compontents. Missing values are allowed.
ncomp	Maximum number of principal components to extract from x.
center	If TRUE, subtract the mean from each column of x before PCA.
scale	if TRUE, divide the standard deviation from each column of x before PCA.
maxiter	Maximum number of NIPALS iterations for each principal component.
tol	Default 1e-6 tolerance for testing convergence of the NIPALS iterations for each principal component.
startcol	Determine the starting column of x for the iterations of each principal component. If 0, use the column of x that has maximum absolute sum. If a number, use that column of x. If a function, apply the function to each column of x and choose the column with the maximum value of the function.
fitted	Default FALSE. If TRUE, return the fitted (reconstructed) value of x.
force.na	Default FALSE. If TRUE, force the function to use the method for missing values, even if there are no missing values in x.
gramschmidt	Default TRUE. If TRUE, perform Gram-Schmidt orthogonalization at each iteration.
verbose	Default FALSE. Use TRUE or 1 to show some diagnostics.

Details

The R2 values that are reported are marginal, not cumulative.

Value

A list with components `eig`, `scores`, `loadings`, `fitted`, `ncomp`, `R2`, `iter`, `center`, `scale`.

Author(s)

Kevin Wright

References

Wold, H. (1966) Estimation of principal components and related models by iterative least squares. In Multivariate Analysis (Ed., P.R. Krishnaiah), Academic Press, NY, 391-420.

Andrecut, Mircea (2009). Parallel GPU implementation of iterative PCA algorithms. Journal of Computational Biology, 16, 1593-1599.

Examples

```
B <- matrix(c(50, 67, 90, 98, 120,
              55, 71, 93, 102, 129,
              65, 76, 95, 105, 134,
              50, 80, 102, 130, 138,
              60, 82, 97, 135, 151,
              65, 89, 106, 137, 153,
              75, 95, 117, 133, 155), ncol=5, byrow=TRUE)
rownames(B) <- c("G1","G2","G3","G4","G5","G6","G7")
colnames(B) <- c("E1","E2","E3","E4","E5")
dim(B) # 7 x 5
p1 <- nipals(B)
dim(p1$scores) # 7 x 5
dim(p1$loadings) # 5 x 5

B2 = B
B2[1,1] = B2[2,2] = NA
p2 = nipals(B2, fitted=TRUE)

# Two ways to make a biplot

# method 1
biplot(p2$scores, p2$loadings)

# method 2
class(p2) <- "princomp"
p2$sdev <- sqrt(p2$eig)
biplot(p2, scale=0)
```

---

uscrime *U.S. Crime rates per 100,00 people*

Description

U.S. Crime rates per 100,00 people for 7 categories in each of the 50 U.S. states in 1977.

Usage

```
uscrime
```

Format

A data frame with 50 observations on the following 8 variables.

state U.S. state

murder murders

rape rapes

robbery robbery

assault assault

burglary burglary

larceny larceny

autotheft automobile thefts

Details

There are two missing values.

Source

Documentation Example 3 for PROC HPPRINCOMP. http://documentation.sas.com/api/docsets/stathpug/14.2/content/stathp

References

SAS/STAT User's Guide: High-Performance Procedures. The HPPRINCOMP Procedure. http://support.sas.com/documentat

Examples

```
library(nipals)
head(uscrime)

# SAS deletes rows with missing values
dat <- uscrime[complete.cases(uscrime), ]
dat <- as.matrix(dat[ , -1])
m1 <- nipals(dat) # complete-data method

# Traditional NIPALS with missing data
dat <- uscrime
dat <- as.matrix(dat[ , -1])
m2 <- nipals(dat, gramschmidt=FALSE) # missing
round(crossprod(m2$loadings),3) # Prin Comps not quite orthogonal

# Gram-Schmidt corrected NIPALS
m3 <- nipals(dat, gramschmidt=TRUE) # TRUE is default
round(crossprod(m3$loadings),3) # Prin Comps are orthogonal
```

Index