Package ‘sGPCA’

February 20, 2015

Type Package

Title Sparse Generalized Principal Component Analysis

Version 1.0

Date 2012-07-05

Author Frederick Campbell

Maintainer Frederick Campbell <frederick.campbell@rice.edu>

Depends Matrix, fields

Description Functions for computing sparse generalized principal components, including functions for modeling structured correlation

License BSD

NeedsCompilation yes

Repository CRAN

Date/Publication 2013-07-06 18:16:30

R topics documented:

gpca ................................................................. 1
Laplacian .......................................................... 3
sGPCA ............................................................. 5
sgpca .............................................................. 5

Index 9

----------

gpca Generalized Principal Component Analysis

----------

Description

Computes the rank K Generalized PCA (GPCA) solution.
Usage

g pca(X, Q, R, K, deflation=FALSE)

Arguments

X
 The n x p data matrix. X must be of class matrix with all numeric values.

Q
 The row generalizing operator, an n x n matrix. Q can be of class matrix or class dcmatrix, but the function is optimized for sparse matrices of class dcmatrix. Q must also be positive semi-definite and be scaled to have operator norm one.

R
 The column generalizing operator, an p x p matrix. R can be of class matrix or class dcmatrix, but the function is optimized for sparse matrices of class dcmatrix. R must also be positive semi-definite and be scaled to have operator norm one.

K
 The number of GPCA components to compute. The default value is one.

deflation
 Algorithm used to calculate the solution. Default is deflation=FALSE and most users should not deviate from this option. See details.

Details

The Generalized PCA solution maximizes the sample variance of the data in an inner-product space induced by the row and column generalizing operators, Q and R, and also finds the best low-rank approximation to the data as measured by a generalization of the Frobenius norm. Note that the resulting GPCA factors U and V are orthogonal with respect to the row and column generalizing operators: U'Q U = I and V'R V = I. Generalized PCA can be interpreted as finding major modes of variation that are independent from the generalizing operators. Thus, if Q and R encode noise structures (see laplacian) or noise covariances (see Exp.cov), then GPCA finds patterns separate from the structure of the noise.

The generalizing operators, Q and R, must be positive semi-definite and have operator norm one. Note that if these are the identity matrix, then GPCA is equivalent to PCA and g pca returns the SVD of X. Smoothers, such as covariances (see Exp.cov, Exp.simple.cov, Rad.cov, stationary.cov, cubic.cov, stationary.tapercov, wendland.cov), and inverse smoothers (see laplacian) can be used as generalizing operators for data in which variables are associated with a specific location (e.g. image data and spatio-temporal data).

This function has the option of using one of two algorithms to compute the solution. The deflation = FALSE option computes the eigen-decomposition of a quadratic form of dimension min(n, p) to find U or V and finds the other factor by regression. The deflation = TRUE option finds each factor using the generalized power algorithm and performs deflation to compute multiple factors. The deflation = FALSE option is generally faster, and especially so when one dimension is much smaller than the other. The option deflation = TRUE is faster only if both dimensions are large n, p > 5,000 and K is small.

Value

U
 The left GPCA factors, an n x K matrix.

V
 The right GPCA factors, an p x K matrix.
Laplacian

D           A vector of the K PCA values.
cumm.prop.var Cumulative proportion of variance explained by the first K components.
prop.var     Proportion of variance explained by each component.

Author(s)

Frederick Campbell

References


See Also

laplacian, Exp.cov.sgPCA

Examples

data(ozone2)
ind = which(apply(is.na(ozone2$y),2,sum)==0)
X = ozone2$y[,ind]
n = nrow(X)
p = ncol(X)

# Generalizing Operators - Spatio-Temporal Smoothers
R = Exp.cov(ozone2$lon.lat[ind,],theta=5)
er = eigen(R,only.values=TRUE);
R = R/max(er$values)
Q = Exp.cov(c(1:n),c(1:n),theta=3)
eq = eigen(Q,only.values=TRUE)
Q = Q/max(eq$values)

# SVD
fitsvd = gpca(X,diag(n),diag(p),1)

# GPCA
fitgpca = gpca(X,Q,R,1)
fitgpca$prop.var # proportion of variance explained

Laplacian

Graph Laplacian Matrices

Description

Functions for generating graph laplacian matrices based on nearest neighbors in a grid structure or neighbors a certain distance apart.
Usage

\begin{verbatim}
laplacian(dims)
distLaplacian(matrix, window=2)
\end{verbatim}

Arguments

- **dims**: A vector containing the dimensions of the underlying grid. One, two and three-dimensional grids are supported.
- **matrix**: A full distance matrix. This can be computed via any distance metric.
- **window**: A scalar denoting the maximum distance at which points are considered neighbors, meaning they share an edge in the graph structure.

Details

Graph Laplacian matrices can be used as generalizing operators in the GPCA framework. These behave like inverse covariances and are useful for recovering edges in the GPCA factors. Laplacians are defined as the degree matrix minus the adjacency matrix for a network structure.

For variables structured as a grid, Laplacians are constructed by placing edges between neighbors in the grid structure. For one dimensional grids (e.g. equally-spaced time points), this is a chain graph. Nearest neighbors for two and three dimensional grids (e.g. image data) are defined using the Chebychev distance. Laplacians for grid structures are scaled to have operator norm one, which is required for use with \texttt{gpca} and \texttt{sgpca}.

Laplacians are constructed from a general distance matrix by placing edges between variables less than or equal to \texttt{window} distance apart. These are returned without scaling to have operator norm one, and must be appropriately scaled before used with \texttt{gpca} and \texttt{sgpca}.

Value

\begin{verbatim}
Q
\end{verbatim}

A sparse matrix of the class \texttt{dgCMatrix}.

Author(s)

Frederick Campbell

Examples

\begin{verbatim}
# Laplacians on a 1D, 2D, and 3D grid
Q = laplacian(c(10))
Q = laplacian(c(10,10))
Q = laplacian(c(10,10,10))

# Laplacians computed based on Euclidean distances
data(ozone2)
D = as.matrix(dist(ozone2$lon.lat))
Q = distLaplacian(D,2)
Q = Q/max(eigen(Q,only.values=TRUE)$values)
\end{verbatim}
Description

Functions for computing Sparse Generalized Principal Components, including functions for modeling structured correlation

Details

Package: sgpca
Type: Package
Version: 1.0
Date: 2012-07-05
License: BSD-3

Author(s)

Frederick Campbell
Maintainer: Frederick Campbell <frederick.campbell@rice.edu>

References


Usage

sgpca(X, Q, R, K = 1, lamu = 0, lamvs = 0, posu = FALSE, posv = FALSE, threshold = 1e-07, maxit = 1000, full.path = FALSE)
Arguments

X  The \( n \times p \) data matrix. \( X \) must be of class \texttt{matrix} with all numeric values.

Q  The row generalizing operator, an \( n \times n \) matrix. \( Q \) can be of class \texttt{matrix} or class \texttt{dcGMatrix}, must be positive semi-definite, and have operator norm one.

R  The column generalizing operator, an \( p \times p \) matrix. \( R \) can be of class \texttt{matrix} or class \texttt{dcGMatrix}, must be positive semi-definite, and have operator norm one.

k  The number of GPCA components to compute. The default value is one.

\( \lambda_\mu \)  The regularization parameter that determines the sparsity level for the row factor, \( U \). The default value is 0. If the data is oriented with rows as samples, non-zero \( \lambda_\mu \) corresponds to two-way sparse methods.

\( \lambda_{mv} \)  A scalar or vector of regularization parameters that determine the sparsity level for the column factor, \( V \). The default is 0, with non-zero values corresponding to sparse or two-way sparse methods. If \( \lambda_{mv} \) is a vector, then the BIC method is used to select the optimal sparsity level. Alternatively, if \texttt{full.path} is specified, then the solution at each value of \( \lambda_{mv} \) is returned.

posu  Flag indicating whether the row factor, \( U \) should be constrained to be strictly positive. The default value is FALSE.

posv  Flag indicating whether the column factor, \( V \) should be constrained to be strictly positive. The default value is FALSE.

threshold  Sets the threshold for convergence. The default value is \( \cdot0001 \).

maxit  Sets the maximum number of iterations. The default value is \( \cdot0001 \).

full.path  Flag indicating whether the entire solution path, or the solution at each value of \( \lambda_{mv} \), should be returned. The default value is FALSE.

Details

The \texttt{sgpca} function has the flexibility to fit combinations of sparsity and/or non-negativity for both the row and column generalized PCs. Regularization is used to encourage sparsity in the GPCA factors by placing an L1 penalty on the GPC loadings, \( V \), and or the sample GPCs, \( U \). Non-negativity constraints on \( V \) and/or \( U \) yield sparse non-negative and two-way non-negative GPCA. Generalizing operators as described for \texttt{gpca} can be used with this function and have the same properties.

When \( \lambda_{mv} = 0 \), \( \lambda_\mu = 0 \), \texttt{posu} = 0, and \texttt{posv} = 0, the GPCA solution also given by \texttt{gpca} is returned. The magnitude of the regularization parameters, \( \lambda_{mv} \) and \( \lambda_\mu \), determine the level of sparsity of the factors \( U \) and \( V \), with higher regularization parameter values yielding sparser factors. If more than one regularization value \( \lambda_{mv} \) is given, then \texttt{sgpca} finds the optimal regularization parameter \( \lambda_{mv} \) by minimizing the BIC derived from the generalized least squares update for each factor.

If \texttt{full.path} = TRUE, then the full path of solutions (\( U \), \( D \), and \( V \)) is returned for each value of \( \lambda_{mv} \) given. This option is best used with 50 or 100 values of \( \lambda_{mv} \) to well approximate the regularization paths. Numerically, the path begins with the GPCA solution, \( \lambda_{mv} = 0 \), and uses warm starts at each step as \( \lambda_{mv} \) increases.

Proximal gradient descent is used to compute each rank-one solution. Multiple components are calculated in a greedy manner via deflation. Each rank-one solution is solved by iteratively fitting generalized least squares problems with penalties or non-negativity constraints. These regression problems are solved by the Iterative Soft-Thresholding Algorithm (ISTA) or projected gradient descent.
Value

- **U** The left sparse GPCA factors, an $n \times K$ matrix. If `full.path` is specified with $r$ values of `lamvs`, then $U$ is a $n \times K \times r$ array.
- **V** The right sparse GPCA factors, a $p \times K$ matrix. If `full.path` is specified with $r$ values of `lamvs`, then $V$ is a $p \times K \times r$ array.
- **D** A vector of the $K$ sparse GPCA values. If `full.path` is specified with $r$ values of `lamvs`, then $D$ is a $K \times r$ matrix.

- **cumulative.prop.var** The cumulative proportion of variance explained by the components
- **bics** The BIC values computed for each value of `lamvs` and each of the $K$ components.
- **optlams** Optimal regularization parameter as chosen by the BIC method for each of the $K$ components.

Author(s)

Frederick Campbell

References


Examples

data(ozone2)
ind = which(apply(is.na(ozone2$y),2,sum)==0)
X = ozone2$y[,ind]
n = nrow(X)
p = ncol(X)

# Generalizing Operators - Spatio-Temporal Smoothers
R = Exp.cov(ozone2$lon.lat[ind],theta=5)
er = eigen(R,only.values=TRUE);
R = R/max(er$values)
Q = Exp.cov(c(1:n),c(1:n),theta=3)
eq = eigen(Q,only.values=TRUE)
Q = Q/max(eq$values)

# Sparse GPCA
fit = sg pca(X,Q,R,K=1,lamu=0,lamvs=c(0.1))
fit$prop.var # proportion of variance explained
fit$optlams # optimal regularization param chosen by BIC
fit$bics # BIC values for each lambda

# Sparse Non-negative GPCA
fit = sg pca(X,Q,R,K=1,lamu=0,lamvs=1, posv=TRUE)

# Two-way Sparse GPCA
fit = sgPCA(X,Q,R,K=1,lambda1=1,lambda2=1)

# Two-way Sparse Non-negative GPCA
fit = sgPCA(X,Q,R,K=1,lambda1=1,lambda2=1,posu=TRUE,posv=TRUE)

# Return full regularization paths for inputted lambda values
fit = sgPCA(X,Q,R,K=1,lambda1=0,lambda2=c(0.1,.5,1),full.path=TRUE)
Index

*Topic **Non-negative Matrix Factorization**
  sgpca, 5
*Topic **Non-negative PCA**
  sgpca, 5
*Topic **PCA**
  gPCA, 1
  sgpca, 5
*Topic **SVD**
  gPCA, 1
*Topic **Sparse PCA**
  sgpca, 5
*Topic **laplacian**
  Laplacian, 3
*Topic **matrix factorization**
  gPCA, 1
  sgpca, 5
*Topic **multivariate analysis**
  gPCA, 1
  sgpca, 5
*Topic **package**
  sGPCA, 5
*Topic **principal components analysis**
  gPCA, 1
  sgpca, 5

cubic.cov, 2
distLaplacian (Laplacian), 3
Exp.cov, 2, 3
Exp.simple.cov, 2

gPCA, 1, 6
Laplacian, 3
laplacian, 2, 3
laplacian (Laplacian), 3
Rad.cov, 2