Package ‘svd’

March 7, 2019

Type Package

Imports methods

Suggests testthat (>= 0.7)

Title Interfaces to Various State-of-Art SVD and Eigensolvers

Version 0.4.2

Author Anton Korobeynikov [aut, cre], Rasmus Munk Larsen [ctb, cph], Lawrence Berkeley National Laboratory [ctb, cph]

Maintainer Anton Korobeynikov <anton@korobeynikov.info>

Description R bindings to SVD and eigensolvers (PROPACK, nuTRLan).

URL http://github.com/asl/svd

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-03-07 22:20:04 UTC

R topics documented:

eigen *Generic Eigendecomposition of a Matrix*

Description

Compute the set of eigenvalues and eigenvectors decomposition of a real or complex rectangular matrix.

Usage

```
trlan.eigen(X, neig = min(m, n), opts = list(), lambda = NULL, U = NULL)
ztrlan.eigen(X, neig = min(m, n), opts = list(), lambda = NULL, U = NULL)
```

Arguments

`X`	the matrix to be decomposed. This can be either normal matrix or 'external matrix' object (e.g. one, created via 'extmat' function).
`neig`	number of desired eigentriples
`opts`	different options for eigensolver. See 'Details' section for more information
`lambda`	set of already computed singular values (used for continuation of the decomposition).
`U`	matrix of already computed eigenvectors (used for continuation of the decomposition).

Details

These routines provides an interface to state-of-art implementation of eigensolver. In particular, nu-TRLAN does the thick-restart Lanczos eigendecomposition of a matrix.

'opts' is a list of different options which can be passed to the routines. Note that by default more or less suitable values for these options are set by the routines automatically.

The options for nu-TRLAN are:

kmax integer, maximum number of iterations.

maxiter integer. maximum number of matrix-vector products.

tol numeric, tolerance level.

verbose integer, verboseness level.

Value

The returned value is a list with components

`d`	a vector containing the eigenvalues of 'X'
`u`	a matrix whose columns contain the eigenvectors of 'X'

References

Wu, K. and Simon, H. (2000). *Thick-restart Lanczos method for large symmetric eigenvalue problems.* SIAM J. Matrix Anal. Appl. 22, 2, 602-616.

Yamazaki, I., Bai, Z., Simon, H., Wang, L.-W., and Wu, K. (2008). *Adaptive projection subspace dimension for the thick restart Lanczos method.* Tech. rep., Lawrence Berkeley National Laboratory, University of California, One Cyclotron road, Berkeley, California 94720.

Korobeynikov, A. (2010) *Computation- and space-efficient implementation of SSA*. Statistics and Its Interface, Vol. 3, No. 3, Pp. 257-268

---

extmat — *External matrices operations.*

Description

A set of routines to operate on "external" matrices.

Usage

```
is.extmat(X)
extmat.ncol(X)
extmat.nrow(X)
extmat(mul, tmul, nrow, ncol, env = parent.frame())
ematmul(emat, v, transposed = FALSE)
```

Arguments

`X, emat`	matrix to operate on
`mul`	function performing the multiplication of matrix to vector
`tmul`	function performing the multiplication of transposed matrix to vector
`nrow`	number of rows of the matrix
`ncol`	number of columns of the matrix
`env`	environment, where matrix-vector multiplication function call is evaluated in
`transposed`	logical, if 'TRUE' the multiplication is performed with the transposed matrix.
`v`	vector to multiply with.

Details

These routines checks whether the given external pointer actually points to "external matrix" structure and allow to extract the number of columns and rows respectively.

'extmat' is a convenient wrapper which allows one provide just the routines which will multiply with matrix and the transposed one (e.g. if the matrix is sparse or structured) and allow to use the SVD routines of the package

Value

Object 'extmat' class

See Also

extmat

Examples

```
## Not run:
library(Matrix)
f <- function(v) as.numeric(A %*% v) # Convert Matrix object back to vector
tf <- function(v) as.numeric(tA %*% v) # Convert Matrix object back to vector

e <- new.env()
assign("A", USCounties, e)
assign("tA", t(USCounties), e)
environment(f) <- e
environment(tf) <- e

m <-extmat(f, tf, nrow(USCounties), ncol(USCounties))
system.time(v1 <- propack.svd(m, neig = 10))
#   user  system elapsed
#  0.252   0.007   0.259
system.time(v2 <- propack.svd(as.matrix(USCounties), neig = 10))
#   user  system elapsed
#  8.563   0.027   8.590

## End(Not run)

# The largest eigenvalue and the corresponding eigenvector of a Hilbert matrix
h <- outer(1:5000, 1:5000, function(i, j) 1 / (i + j - 1))
v1 <- trlan.eigen(h, neig = 1)
# Same, but using complex numbers
v2 <- ztrlan.eigen(h, neig = 1)
print(c(v1$d, v2$d))
```

---

extmat-class *Class* "extmat"

Description

'extmat' is a convenient wrapper which allows one provide just the routines which will multiply with matrix and the transposed one (e.g. if the matrix is sparse or structured) and allow to use the SVD routines of the package. This S4 wrapper allows the use of usual matrix operations on such objects.

Objects from the Class

Objects can be created by calls of the form `extmat(mul, tmul, nrow, ncol, env = parent.frame())`.

See Also

extmat

Examples

```
## Not run:
library(Matrix)
f <- function(v) as.numeric(A %*% v) # Convert Matrix object back to vector
tf <- function(v) as.numeric(tA %*% v) # Convert Matrix object back to vector

e <- new.env()
assign("A", USCounties, e)
assign("tA", t(USCounties), e)
environment(f) <- e
environment(tf) <- e

m <-extmat(f, tf, nrow(USCounties), ncol(USCounties))
system.time(v1 <- propack.svd(m, neig = 10))
#   user  system elapsed
#  0.252   0.007   0.259
system.time(v2 <- propack.svd(as.matrix(USCounties), neig = 10))
#   user  system elapsed
#  8.563   0.027   8.590

## End(Not run)
```

---

svd *Generic Singular Value Decomposition of a Matrix*

Description

Compute the singular-value decomposition of a real or complex rectangular matrix.

Usage

```
propack.svd(X, neig = min(m, n), opts = list())
trlan.svd(X, neig = min(m, n), opts = list(), lambda = NULL, U = NULL)
ztrlan.svd(X, neig = min(m, n), opts = list(), lambda = NULL, U = NULL)
```

Arguments

X	the matrix to be decomposed. This can be either normal matrix or 'external matrix' object (e.g. one, created via 'extmat' function).
neig	number of desired eigentriples
opts	different options for eigensolver. See 'Details' section for more information
lambda	set of already computed singular values (used for continuation of the decomposition).
U	matrix of already computed eigenvectors (used for continuation of the decomposition).

Details

These routines provides an interface to two state-of-art implementations of truncated SVD.

PROPACK does this via the implicitly restarted Lanczos bidiagonalization with partial reorthogonalization. nu-TRLAN does the thick-restart Lanczos eigendecomposition of cross-product matrix.

'opts' is a list of different options which can be passed to the routines. Note that by default more or less suitable values for these options are set by the routines automatically.

The options for PROPACK are:

kmax integer, maximum number of iterations.

dim integer, dimension of Krylov subspace.

p integer, number of shifts per restart.

maxiter integer. maximum number of restarts.

tol numeric, tolerance level.

verbose logical, if 'TRUE', provide verbose output.

The options for nu-TRLAN are:

kmax integer, maximum number of iterations.

maxiter integer. maximum number of matrix-vector products.

tol numeric, tolerance level.

verbose integer, verboseness level.

Value

The returned value is a list with components

d	a vector containing the singular values of 'x'
u	a matrix whose columns contain the left singular vectors of 'X'
v	a matrix whose columns contain the right singular vectors of 'X' (only for 'propack.svd')

References

Wu, K. and Simon, H. (2000). *Thick-restart Lanczos method for large symmetric eigenvalue problems.* SIAM J. Matrix Anal. Appl. 22, 2, 602-616.

Yamazaki, I., Bai, Z., Simon, H., Wang, L.-W., and Wu, K. (2008). *Adaptive projection subspace dimension for the thick restart Lanczos method.* Tech. rep., Lawrence Berkeley National Laboratory, University of California, One Cyclotron road, Berkeley, California 94720.

Larsen, R. M. (1998). *Efficient algorithms for helioseismic inversion.* Ph.D. thesis, University of Aarhus, Denmark.

Korobeynikov, A. (2010) *Computation- and space-efficient implementation of SSA.* Statistics and Its Interface, Vol. 3, No. 3, Pp. 257-268

Index