Package ‘ssvd’

February 20, 2015

Type Package

Title Sparse SVD

Version 1.0

Date 2013-09-25

Author Dan Yang

Maintainer Dan Yang <dyang@stat.rutgers.edu>

Description Fast iterative thresholding sparse SVD, together with an initialization algorithm

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2013-09-26 17:23:50

R topics documented:

`ssvd-package` *Sparse SVD*

Description

Obtain sparse SVD using fast iterative thresholding method, together with a fast initialization algorithm

Details

Package:	ssvd
Type:	Package
Version:	1.0
Date:	2013-09-25
License:	GPL (>= 2)

There are three main functions of the package: ssvd, ssvd.initial, and ssvd.iter.thresh

Author(s)

Dan Yang <dyang@stat.rutgers.edu>

References

A Sparse SVD Method for High-dimensional Data

Examples

```
ssvd(matrix(rnorm(2^15),2^7,2^8), method = "method")
ans.initial <- ssvd.initial(matrix(rnorm(2^15),2^7,2^8), method = "method")
ans.iter <- ssvd.iter.thresh(matrix(rnorm(2^15),2^7,2^8),
u.old=ans.initial$u, v.old= ans.initial$v, method = "method")
```

---

ssvd *Sparse SVD*

Description

The function ssvd combines two functions ssvd.initial and ssvd.iter.thresh into one. Obtain sparse SVD using fast iterative thresholding method, together with a fast initialization algorithm

Usage

```
ssvd(x, method = c("theory", "method"), alpha.method = 0.05, alpha.theory = 1.5,
huber.beta = 0.95, sigma = NA, r = 1, gamma.u = sqrt(2), gamma.v = sqrt(2),
dothres = "hard", tol = 1e-08, n.iter = 100, n.boot = 100, non.orth = FALSE)
```

Arguments

x	Input matrix, for which one would like to get a sparse SVD.
method	If method = "theory", then a theoretical procedure is adopted which is based on normal assumption on the noise. If method = "method", then the function bypass the normal assumption by some robust statistics. These two choices typically give similar solutions, but "theory" is much faster.

`alpha.method`	Alpha.method is the level of the hypothesis test when one performs Holm multiple hypothesis testing, which is used to select the candidate rows and columns in the initialization step. The value is usually set to be 0.05.
`alpha.theory`	Alpha.theory is a scaler that is used when normal assumption is true, method="theory", and a chisq tail bound is used to select the candidate rows and columns in the initialization step. Most of the time, users should keep it as it is.
`huber.beta`	Huber.beta is a scaler which is the cut-off point in the Huber function. The huberization is utilized to achieve robustness when normal assumption is violated in the initialization step.
`sigma`	Sigma is a scaler for the noise level. The user can set it to be NA, and the function will estimate it automatically.
`r`	A scaler, the number of components, i.e., the number of singular vectors to be computed.
`gamma.u`	When the method="theory", gamma.u=sqrt(2) corresponds to the sqrt(2 log(p)), which is the largest magnitude of p iid standard normals. If gamma.u is manually set to be smaller or larger than sqrt2, the left singular vectors will be denser or sparser respectively.
`gamma.v`	When the method="theory", gamma.u=sqrt(2) corresponds to the sqrt(2 log(p)), which is the largest magnitude of p iid standard normals. If gamma.u is manually set to be smaller or larger than sqrt2, the right singular vectors will be denser or sparser respectively.
`dothres`	Dothres has two choices, either "hard" or "soft", which means hard-thresholding or soft-thresholding.
`tol`	The tolerance level that determines when the algorithm stops.
`n.iter`	Maximum number of iterations allowed.
`n.boot`	Number of bootstrap to estimate the threshold level when method = "method"
`non.orth`	If non.orth=TRUE, then the last iteration of the algorithm will not involve orthoganolization, which should produce sparse solutions.

Value

`u`	A matrix containing left singular vectors
`v`	A matrix containing right singular vectors
`d`	A vector containing singular values
`niter`	Number of iterations for the algorithm to converge
`sigma.hat`	An estimate of the noise level
`dist.u`	The distance between the left singular vectors of the last two iterations, can be used to see whether the algorithm indeed converges.
`dist.v`	The distance between the right singular vectors of the last two iterations, can be used to see whether the algorithm indeed converges.

Author(s)

Dan Yang

References

A Sparse SVD Method for High-dimensional Data

See Also

ssvd.initial and ssvd.iter.thresh

Examples

```
ssvd(matrix(rnorm(2^15),2^7,2^8), method = "method")
```

`ssvd.initial` *This function is used to initialize the sparse SVD iterative method.*

Description

This function is used to initialize the sparse SVD iterative method. The function selects some rows and columns of the input matrix and perform regular SVD of the reduced the matrix with only selected rows and columns, which gives an initial solution to the sparse SVD problem.

Usage

```
ssvd.initial(x, method = c("theory", "method"), alpha.method = 0.05,
alpha.theory = 1.5, huber.beta = 0.95, sigma = NA, r = 1)
```

Arguments

`x`	Input matrix, for which one would like to get a sparse SVD.
`method`	If method = "theory", then a theoretical procedure is adopted which is based on normal assumption on the noise. If method = "method", then the function bypass the normal assumption by some robust statistics. These two choices typically give similar solutions, but "theory" is much faster.
`alpha.method`	Alpha.method is the level of the hypothesis test when one performs Holm multiple hypothesis testing, which is used to select the candidate rows and columns. The value is usually set to be 0.05.
`alpha.theory`	Alpha.theory is a scaler that is used when normal assumption is true, method="theory", and a chisq tail bound is used to select the candidate rows and columns. Most of the time, users should keep it as it is.
`huber.beta`	Huber.beta is a scaler which is the cut-off point in the Huber function. The huberization is utilized to achieve robustness when normal assumption is violated.
`sigma`	Sigma is a scaler for the noise level. The user can set it to be NA, and the function will estimate it automatically.
`r`	A scaler, the number of components, i.e., the number of singular vectors to be computed.

Value

u	A matrix containing left singular vectors
v	A matrix containing right singular vectors
d	A vector containing singular values
sigma.hat	An estimate of the noise level

Author(s)

Dan Yang

References

A Sparse SVD Method for High-dimensional Data

Examples

```
ssvd.initial(matrix(rnorm(2^15),2^7,2^8), method = "method")
```

---

ssvd.iter.thresh *Iterative thresholding sparse SVD*

Description

The function computes sparse SVD by iterative thresholding algorithm with an initializtion as one of the inputs

Usage

```
ssvd.iter.thresh(x, method = c("theory", "method"), u.old, v.old,
gamma.u = sqrt(2), gamma.v = sqrt(2), dothres = "hard", r = ncol(u.old),
tol = 1e-08, n.iter = 100, n.boot = 100, sigma = NA, non.orth = FALSE)
```

Arguments

x	Input matrix, for which one would like to get a sparse SVD.
method	If method = "theory", then a theoretical procedure is adopted which is based on normal assumption on the noise. If method = "method", then the function bypass the normal assumption by some robust statistics. These two choices typically give similar solutions, but "theory" is much faster.
u.old	A matrix that contains initial left singular vectors as the columns of the matrix.
v.old	A matrix that contains initial right singular vectors as the columns of the matrix.
gamma.u	When the method="theory", gamma.u=sqrt(2) corresponds to the sqrt(2 log(p)), which is the largest magnitude of p iid standard normals. If gamma.u is manually set to be smaller or larger than sqrt2, the left singular vectors will be denser or sparser respectively.

gamma.v	When the method="theory", gamma.u=sqrt(2) corresponds to the sqrt(2 log(p)), which is the largest magnitude of p iid standard normals. If gamma.u is manually set to be smaller or larger than sqrt2, the right singular vectors will be denser or sparser respectively.
dothres	Dothres has two choices, either "hard" or "soft", which means hard-thresholding or soft-thresholding
r	A scaler, the number of components, i.e., the number of singular vectors to be computed.
tol	The tolerance level that determines when the algorithm stops.
n.iter	Maximum number of iterations allowed.
n.boot	Number of bootstrap to estimate the threshold level when method = "method"
sigma	Sigma is a scaler for the noise level. The user can set it to be NA, and the function will estimate it automatically.
non.orth	If non.orth=TRUE, then the last iteration of the algorithm will not involve orthoganolization, which should produce sparse solutions.

Value

u	A matrix containing left singular vectors
v	A matrix containing right singular vectors
d	A vector containing singular values
niter	Number of iterations for the algorithm to converge
sigma.hat	An estimate of the noise level
dist.u	The distance between the left singular vectors of the last two iterations, can be used to see whether the algorithm indeed converges.
dist.v	The distance between the right singular vectors of the last two iterations, can be used to see whether the algorithm indeed converges.

Author(s)

Dan Yang

References

A Sparse SVD Method for High-dimensional Data

Examples

```
ans.initial <- ssvd.initial(matrix(rnorm(2^15),2^7,2^8), method = "method")
ans.iter <- ssvd.iter.thresh(matrix(rnorm(2^15),2^7,2^8),
u.old=ans.initial$u, v.old= ans.initial$v, method = "method")
```

Index