Package ‘RobRSVD’

February 19, 2015

Type Package

Title Robust Regularized Singular Value Decomposition

Version 1.0

Date 2013-12-15

Author Lingsong Zhang and Chao Pan

Maintainer Lingsong Zhang <lingsong@purdue.edu>

Description This package provides the function to calculate SVD, regularized SVD, robust SVD and robust regularized SVD method. The robust SVD methods use alternating iteratively reweighted least squares methods. The regularized SVD uses generalized cross validation to choose the optimal smoothing parameters.

License GPL

NeedsCompilation no

Repository CRAN

Date/Publication 2013-12-16 17:46:34

R topics documented:

`RobRSVD-package` *The Robust Regularized Singular Value Decomposition Package*

Description

This package provides the function to calculate SVD, regularized SVD, robust SVD and robust regularized SVD method. The robust SVD methods use alternating iteratively reweighted least squares methods. The regularized SVD uses generalized cross validation to choose the optimal smoothing parameters.

Details

Package:	RobRSVD
Type:	Package
Version:	1.0
Date:	2013-12-15
License:	GPL

The most important function in this package is RobRSVD

Author(s)

Authors are Lingsong Zhang (lingsong@purdue.edu) and Chao Pan (panc@purdue.edu) Maintainer: Lingsong Zhang <lingsong@purdue.edu>

References

Zhang, L., Shen, H., & Huang, J. Z. (2013). Robust regularized singular value decomposition with application to mortality data. The Annals of Applied Statistics, 7(3), 1540-1561.

See Also

See also in svd3dplot

Examples

```
#generate a simulated data set, which is provided in Zhang et al (2013) AoAS paper.
u0<-log(10/9)*10^seq(0, 1, length=100)
v0<-sin(2*pi*seq(0, 1, length=100))/(1+1/pi)
s0<-773
data0<-s0*u0 %*% t(v0)
data<-data0+matrix(rnorm(10000, sd=20), nrow=100)
data[ceiling(10000*runif(50))]<-max(data0)+max(data0)*runif(50)
#the above provides random outlying cell simulation

#the svd calculation
data.svd<-RobRSVD(data, irobust=FALSE, uspar=0, vspar=0)

#the robsvd calculation
data.robsvd<-RobRSVD(data, irobust=TRUE, uspar=0, vspar=0)

#the ssvd calculation
data.ssvd<-RobRSVD(data, irobust=FALSE, iugcv=TRUE, ivgcv=TRUE)

#the robrsvd calculation
data.robrsvd<-RobRSVD(data, irobust=TRUE, iugcv=TRUE, ivgcv=TRUE)

#compare v's
plot(data.svd$v, type='l', ylab='V')
lines(data.robrsvd$v, col=2)
```

```
lines(data.ssvd$v, col=6)
lines(data.robsvd$v, col=4)

#compare u's
plot(data.svd$u, type='l', ylab='U')
lines(data.robrsvd$u, col=2)
lines(data.ssvd$u, col=6)
lines(data.robsvd$u, col=4)

#compare approximation matrices
#app.svd=data.svd$s * data.svd$u %*% t(data.svd$v)
#app.ssvd=data.ssvd$s * data.ssvd$u %*% t(data.ssvd$v)
#app.robsvd=data.robsvd$s * data.robsvd$u %*% t(data.robsvd$v)
#app.robrsvd=data.robrsvd$s * data.robrsvd$u %*% t(data.robrsvd$v)

#par(mfrow=c(2, 2))
#persp(app.svd, main='SVD', theta=-45, phi=40, xlab='', ylab='', zlab='')
#persp(app.ssvd, main='Regularized SVD', theta=-45, phi=40, xlab='', ylab='', zlab='');
#persp(app.robsvd, main='Robust SVD', theta=-45, phi=40, xlab='', ylab='', zlab='');
#persp(app.robrsvd, main='RobRSVD', theta=-45, phi=40, xlab='', ylab='', zlab='');
#dev.off()
```

---

huberWeightLS *Huber's function*

Description

This function provides the usual Huber's weight function in Robust estimation context. See Huber (1981) for details. Let $\rho(x)$ be the usual Huber's function, this function is $\rho'(x)/x$, where $\rho'(x)$ is the derivative of $\rho(x)$.

Usage

```
huberWeightLS(data, k)
```

Arguments

`data`	The input vector or matrix
`k`	The parameter to control robustness, the default value is 1.345, as suggested in many Robust Statistics textbooks

Details

See details of the huber weights in iterative least squares in the references

Value

the weight vector or matrix for Huber-type robust regression

Author(s)

Lingsong Zhang (lingsong@purdue.edu) and Chao Pan (panc@purdue.edu)

References

Huber, P. J. (1981). Robust statistics. Wiley series in probability and mathematical statistics.

Examples

```
#generate a t distrbution matrix
x<-matrix(rt(100, 1), nrow=10)

#generate the huber weight matrix with k=1.345
y=huberWeightLS(x, k=1.345)
```

RobRSVD *Robust Regularized Singular Value Decomposition*

Description

This function provides the Robust Regularized Singular Value Decomposition method based on Zhang, Shen and Huang (2013). We will return the first triplets: singular value, left and right singular vectors, for the first robust and regualrized SVD component.

Usage

```
RobRSVD(data, irobust = F, huberk = 1.345, iinituv = F, inits, initu, initv,
niter = 1000, tol = 1e-05, istablize = T, uspar = 0, vspar = 0, iugcv = F,
ivgcv = F, usparmax = 10000, usparmin = 1e-10, nuspar = 14, iloguspar = T,
vsparmax = 10000, vsparmin = 1e-10, nvspar = 14, ilogvspar = T)
```

Arguments

`data`	The input data.
`irobust`	A logical value. `TRUE` means a robust method is used. `FALSE` (default) means non-robust method is used.
`huberk`	The Huber robustness parameter. The default value is 1.345, as suggested in many Robust Statistics textbook
`iinituv`	A logical value. `TRUE` means initial value of `s`, `u`, and `v` will be provided.
`inits`	The initial value for `s`
`initu`	The initial value for `u`
`initv`	The initial value for `v`
`niter`	The largest possible iteration number. The default value is set to be 1000.
`tol`	The tolerance for numberical zero. The default value is 1e-5

istablize	A logical value. TRUE means that before applying RobRSVD method, we will normalized the data. FALSE means that no normalization will be used.
uspar	A specified smoothing parameter for u
vspar	A specified smoothing parameter for v
iugcv	A logical value. TRUE means that the program will use GCV to choose optimal smoothing parameter for u. Otherwise, it will either use 0 or the parameter specified in uspar.
ivgcv	A logical value. TRUE means that the program will use GCV to choose optimal smoothing parameter for v. Otherwise, it will either use 0 or the parameter specified in vspar.
usparmax	When iugcv is TRUE, this one is to specify the largest possible smoothing parameter for u.
usparmin	When iugcv is TRUE, this one is to specify the smallest possible smoothing parameter for u.
nuspar	When iugcv is TRUE, this one is to specify number of possible smoothing parameters for u.
iloguspar	A logical value. When iugcv is TRUE, this one is to specify whether the equal spaced interval is defined in log-scale (if TRUE) or the original scale (if FALSE), for u.
vsparmax	When ivgcv is TRUE, this one is to specify the largest possible smoothing parameter for v.
vsparmin	When ivgcv is TRUE, this one is to specify the smallest possible smoothing parameter for v.
nvspar	When ivgcv is TRUE, this one is to specify number of possible smoothing parameters for v.
ilogvspar	A logical value. When ivgcv is TRUE, this one is to specify whether the equal spaced interval is defined in log-scale (if TRUE) or the original scale (if FALSE), for v.

Details

This program applied alternating regression technique to estimate singular value decomposition. The usual least squares for regular SVD is replaced by the iterative reweighted least squares to achieve robustness.

Value

A list contains the following fields

s	The singular value
u	The left singular vector, or singular column
v	The right singular vector, or singular row
diagout	A list of diagnosis measures, which include ugcvscore, vgcvscore, ugcvmat and vgcvmat

Author(s)

Lingsong Zhang (lingsong@purdue.edu) and Chao Pan (panc@purdue.edu)

References

Zhang, L., Shen, H., & Huang, J. Z. (2013). Robust regularized singular value decomposition with application to mortality data. The Annals of Applied Statistics, 7(3), 1540-1561.

See Also

See Also as svd, svd3dplot

Examples

```
#generate a simulated data set, which is provided in Zhang et al (2013) AoAS paper.
u0<-log(10/9)*10^seq(0, 1, length=100)
v0<-sin(2*pi*seq(0, 1, length=100))/(1+1/pi)
s0<-773
data0<-s0*u0 %*% t(v0)
data<-data0+matrix(rnorm(10000, sd=20), nrow=100)
data[ceiling(10000*runif(50))]<-max(data0)+max(data0)*runif(50)
#the above provides random outlying cell simulation

#the svd calculation
data.svd<-RobRSVD(data, irobust=FALSE, uspar=0, vspar=0)

#the robsvd calculation
data.robsvd<-RobRSVD(data, irobust=TRUE, uspar=0, vspar=0)

#the ssvd calculation
data.ssvd<-RobRSVD(data, irobust=FALSE, iugcv=TRUE, ivgcv=TRUE)

#the robrsvd calculation
data.robrsvd<-RobRSVD(data, irobust=TRUE, iugcv=TRUE, ivgcv=TRUE)

#compare v's
plot(data.svd$v, type='l', ylab='V')
lines(data.robrsvd$v, col=2)
lines(data.ssvd$v, col=6)
lines(data.robsvd$v, col=4)

#compare u's
plot(data.svd$u, type='l', ylab='U')
lines(data.robrsvd$u, col=2)
lines(data.ssvd$u, col=6)
lines(data.robsvd$u, col=4)

#compare approximation matrices
#app.svd=data.svd$s * data.svd$u %*% t(data.svd$v)
#app.ssvd=data.ssvd$s * data.ssvd$u %*% t(data.ssvd$v)
#app.robsvd=data.robsvd$s * data.robsvd$u %*% t(data.robsvd$v)
#app.robrsvd=data.robrsvd$s * data.robrsvd$u %*% t(data.robrsvd$v)
```

```
#par(mfrow=c(2, 2))
#persp(app.svd, main='SVD', theta=-45, phi=40, xlab='', ylab='', zlab='')
#persp(app.ssvd, main='Regularized SVD', theta=-45, phi=40, xlab='', ylab='', zlab='');
#persp(app.robsvd, main='Robust SVD', theta=-45, phi=40, xlab='', ylab='', zlab='');
#persp(app.robrsvd, main='RobRSVD', theta=-45, phi=40, xlab='', ylab='', zlab='');
#dev.off()
```

---

ssmatls *The smoothing spline matrix*

Description

This function returns the smoothing matrix based on cubic smoothing spline method

Usage

```
ssmatls(n)
```

Arguments

n	the number of (equally spaced) grid

Details

This function is based on Green and Silverman (1994) smoothing spline technique

Value

A smoothing matrix based on smoothing spline method

Author(s)

Lingsong Zhang (lingsong@purdue.edu) Chao Pan (panc@purdue.edu)

References

Green, P. J., & Silverman, B. W. (1994). Nonparametric regression and generalized linear models: a roughness penalty approach (pp. 12-27). London: Chapman & Hall.

See Also

See Also as `smooth.spline`

Examples

```
#set the number of grid
n<-100

#calculate the smoothing matrix
g<-ssmatls(n)
```

Index