Package ‘SplitKnockoff’

January 20, 2025

Type Package

Title Split Knockoffs for Structural Sparsity

Version 2.1

Date 2024-10-13

Author Yuxuan Chen [aut, cre] (Development of the latest version of the packages),
Haoxue Wang [aut] (Development of the first version of the packages),
Yang Cao [aut] (Revison of this package),
Xinwei Sun [aut] (Original ideas about the package),
Yuan Yao [aut] (Testing for the package and management of the development)

Maintainer Yuxuan Chen <yx.chen@connect.ust.hk>

Description Split Knockoff is a data adaptive variable selection framework for controlling the (directional) false discovery rate (FDR) in structural sparsity, where variable selection on linear transformation of parameters is of concern. This proposed scheme relaxes the linear subspace constraint to its neighborhood, often known as variable splitting in optimization.
Simulation experiments can be reproduced following the Vignette.
'Split Knockoffs' is first defined in Cao et al. (2021) <doi:10.48550/arXiv.2103.16159>.

Encoding UTF-8

RoxygenNote 7.2.3

License MIT + file LICENSE

Depends R (>= 3.5.0)

Imports glmnet, MASS, latex2exp, RSpectra, ggplot2, Matrix, stats, mvtnorm

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2024-10-14 15:00:06 UTC

Contents

`canonicalSVD` *singular value decomposition*

Description

Computes a reduced SVD without sign ambiguity. Our convention is that the sign of each vector in U is chosen such that the coefficient with largest absolute value is positive.

Usage

```
canonicalSVD(X)
```

Arguments

`X`	the input matrix

Value

S

U

V

Examples

```
nu = 10
n = 350
m = 100
A_gamma <- rbind(matrix(0,n,m),-diag(m)/sqrt(nu))
svd.result = canonicalSVD(A_gamma)
S <- svd.result$S
S <- diag(S)
V <- svd.result$V
```

cv_all	*calculate the CV optimal beta*

Description

cv_all calculate the CV optimal beta in the problem 1/n |y - X beta|^2 + 1/nu |D beta - gamma|^2 + lambda |gamma|_1.

Usage

```
cv_all(X, y, D, option)
```

Arguments

X	the design matrix
y	the response vector
D	the linear transform
option	options for screening

Value

beta_hat: CV optimal beta

stat_cv: various intermedia statistics

cv_screen	*calculate the CV optimal beta and estimated support set*

Description

cv_all calculate the CV optimal beta and estimated support set in the problem 1/n |y - X beta|^2 + 1/nu |D beta - gamma|^2 + lambda |gamma|_1.

Usage

```
cv_screen(X, y, D, option)
```

Arguments

X	the design matrix
y	the response vector
D	the linear transform
option	options for screening

Value

beta_hat: CV optimal beta

stat_cv: various intermedia statistics, including the estimated support sets

---

`hittingpoint` *hitting point calculator on a given path*

Description

calculate the hitting time and the sign of respective variable in a path.

Usage

```
hittingpoint(coef, lambdas)
```

Arguments

`coef`	the path for one variable
`lambdas`	respective value of lambda in the path

Value

Z: the hitting time

r: the sign of respective variable at the hitting time

`normc` *default normalization function for matrix*

Description

normalize columns of a matrix.

Usage

```
normc(X)
```

Arguments

`X`	the input martix

Value

Y: the output matrix

Examples

```
library(mvtnorm)
n = 350
p = 100
Sigma = matrix(0, p, p)
X <- rmvnorm(n,matrix(0, p, 1), Sigma)
X <- normc(X)
```

---

select — *split knockoff selector given W statistics*

Description

split knockoff selector given W statistics

Usage

```
select(W, q, method = "knockoff+")
```

Arguments

W	statistics W_j for testing null hypothesis
q	target FDR
method	option$method can be 'knockoff' or 'knockoff+'

Value

S: array of selected variable indices

---

sk.create — *generate split knockoff copies*

Description

Gives the variable splitting design matrix and response vector. It will also create a split knockoff copy if required.

Usage

```
sk.create(X, y, D, nu, option)
```

Arguments

X	the design matrix
y	the response vector
D	the linear transform
nu	the parameter for variable splitting
option	options for creating the Knockoff copy option$copy true : create a knockoff copy;

Value

A_beta: the design matrix for beta after variable splitting

A_gamma: the design matrix for gamma after variable splitting

tilde_y: the response vector after variable splitting.

tilde_A_gamma: the knockoff copy of A_beta; will be NULL if option$copy = false.

Examples

```
option <- list()
option$q <- 0.2
option$method <- 'knockoff'
option$normalize <- 'true'
option$lambda <- 10.^seq(0, -6, by=-0.01)
option$nu <- 10
option$copy <- 'true'
library(mvtnorm)
sigma <-1
p <- 100
D <- diag(p)
m <- nrow(D)
n <- 350
nu = 10
c = 0.5
Sigma = matrix(0, p, p)
for( i in 1: p){
  for(j in 1: p){
    Sigma[i, j] <- c^(abs(i - j))
 }
}
X <- rmvnorm(n,matrix(0, p, 1), Sigma)
beta_true <- matrix(0, p, 1)
varepsilon <- rnorm(n) * sqrt(sigma)
y <- X %*% beta_true + varepsilon
creat.result  <- sk.create(X, y, D, nu, option)
A_beta  <- creat.result$A_beta
A_gamma <- creat.result$A_gamma
tilde_y <- creat.result$tilde_y
tilde_A_gamma <- creat.result$tilde_A_gamma
```

`sk.decompose` *make SVD as well as orthogonal complements*

Description

make SVD as well as orthogonal complements

Usage

```
sk.decompose(A, D)
```

Arguments

`A`	the input matrix
`D`	the linear transform

Value

U

S

V

U_perp: orthogonal complement for U

Examples

```
library(mvtnorm)
n = 350
p = 100
D <- diag(p)
Sigma = matrix(0, p, p)
X <- rmvnorm(n,matrix(0, p, 1), Sigma)
decompose.result <- sk.decompose(X, D)
U_perp <- decompose.result$U_perp
```

`sk.filter` *split Knockoff filter for structural sparsity problem*

Description

split Knockoff filter for structural sparsity problem

Usage

```
sk.filter(X, D, y, option)
```

Arguments

`X`	the design matrix
`D`	the response vector
`y`	the linear transformation
`option`	options for creating the Split Knockoff statistics. option$q: the desired FDR control target. option$beta: choices on beta(lambda), can be: 'path', beta(lambda) is taken from a regularization path; 'cv_beta', beta(lambda) is taken as the cross validation optimal estimator hat beta; or 'cv_all', beta(lambda) as well as nu are taken from the cross validation optimal estimators hat beta and hat nu.The default setting is 'cv_all'. option$lambda_cv: a set of lambda appointed for cross validation in estimating hat beta, default 10.^seq(0, -8, by = -0.4). option$nu_cv: a set of nu appointed for cross validation in estimating hat beta and hat nu, default 10.^seq(0, 2, by = 0.4). option$nu: a set of nu used in option.beta = 'path' or 'cv_beta' for Split Knockoffs, default 10.^seq(0, 2, by = 0.2). option$lambda: a set of lambda appointed for Split LASSO path calculation, default 10.^seq(0, -6, by = -0.01). option$normalize: whether to normalize the data, default true. option$W: the W statistics used for Split Knockoffs, can be 's', 'st', 'bc', 'bct', default 'st'.

Value

various intermedia statistics

---

sk.W_fixed *W statistics generator based on a fixed beta(lambda) = hat beta*

Description

generates the split knockoff statistics W based on a fixed beta(lambda) = hat beta in the intercepetion assignment step.

Usage

```
sk.W_fixed(X, D, y, nu, option)
```

Arguments

`X`	the design matrix
`D`	the linear transform
`y`	the response vector
`nu`	the parameter for variable splitting
`option`	options for creating the Knockoff statistics option$lambda: the choice of lambda for the path option$beta_hat: the choice of beta(lambda) = hat beta

Value

the split knockoff statistics W and various intermedia statistics

---

`sk.W_path` *W statistics generator based on the beta(lambda) from a split LASSO path*

Description

generates the split knockoff statistics W based on the beta(lambda) from a split LASSO path in the intercepetion assignment step.

Usage

```
sk.W_path(X, D, y, nu, option)
```

Arguments

`X`	the design matrix
`D`	the linear transform
`y`	the response vector
`nu`	the parameter for variable splitting
`option`	options for creating the Knockoff statistics option$lambda: the choice of lambda for the path

Value

the split knockoff statistics W and various intermedia statistics

---

`threshold` *compute the threshold for variable selection*

Description

compute the threshold for variable selection

Usage

```
threshold(W, q, method = "knockoff+")
```

Arguments

`W`	statistics W_j for testing null hypothesis beta_j = 0
`q`	target FDR
`method`	option$method can be 'knockoff' or 'knockoff+'

Value

T: threshold for variable selection

Index