Package 'SplitKnockoff'

September 13, 2021

Type Package

Title Split Knockoffs for Structural Sparsity

Version 0.8

Date 2021-09-01

Author Haoxue Wang [aut, cre] (Development of the whole 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 Haoxue Wang <haoxwang@student.ethz.ch>

Description A novel method for controlling the false discovery rate (FDR) in structural sparsity setting. 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 defined in Cao et al. (2021) <arXiv:2103.16159>.

URL https://github.com/wanghaoxue0/SplitKnockoff

BugReports https://github.com/wanghaoxue0/SplitKnockoff/issues

Depends R (>= 3.5.0)

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

Suggests knitr, rmarkdown

Encoding UTF-8

VignetteBuilder knitr

NeedsCompilation no

RoxygenNote 7.1.1

License MIT + file LICENSE

Repository CRAN

Date/Publication 2021-09-13 12:40:02 UTC

R topics documented:

`canonicalSVD` *canonicalSVD*

Description

Computes a reduced SVD without sign ambiguity

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_filter *cv_filter*

Description

Split Knockoff filter for structural sparsity problem, using cross validation

Usage

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

Arguments

X	the design matrix
D	the response vector
y	the linear transform
option	options for creating the Knockoff statistics option$eta the choice of eta for creating the knockoff copy option$q the desired FDR control bound option$method 'knockoff' or 'knockoff+' option$stage0 choose the method to conduct split knockoff 'fixed': fixed intercept assignment for PATH ORDER method. option$beta : the choice of fixed beta for step 0: 'mle': maximum likelihood estimator. 'ridge': ridge regression choice beta with lambda = 1/nu. 'cv_split': cross validation choice of split LASSO over nu and lambda. 'cv_ridge': cross validation choice of ridge regression over lambda. 'path': take the regularization path of split LASSO as the intercept assignment for PATH ORDER method. 'magnitude': using MAGNITUDE method. option$lambda: a set of lambda appointed for path calculation option$nu: a set of nu used for Split Knockoffs option$k_fold: the fold used in cross validation option$cv_rule: the rule used in CV 'min': choose nu with minimal CV loss. 'complexity': choose nu with minimal model complexity in the range of 0.99 * CV_loss <= min(CV_loss).

Value

result: selected features of Split Knockoffs with CV optimal selection of nu.

CV_loss: the CV loss of Split Knockoffs w.r.t. nu.

nu_optimal: the CV optimal nu.

normc *normc*

Description

Normalize columns of a matrix. A clone of NORMC from the Neural Network toolbox.

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)
```

---

private.hittingpoint *private.hittingpoint*

Description

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

Usage

```
private.hittingpoint(coef, lambda_vec)
```

Arguments

coef	the path for one variable
lambda_vec	respective value of lambda in the path

Value

Z: the hitting time

r: the sign of respective variable at the hitting time

simu_eval *simu_eval*

Description

calculate the FDR and Power when gamma^* is available, i.e. in simulations.

Usage

```
simu_eval(gamma_true, result)
```

Arguments

gamma_true	true signal of gamma
result	the estimated support set of gamma

Value

fdr: false discovery rate of the estimated support set

power: power of the estimated support set

simu_unit *simu_unit*

Description

the simulation unit for simulation experiments.

Usage

```
simu_unit(n, p, D, A, c, k, option)
```

Arguments

n	the sample size
p	the dimension of variables
D	the linear transform
A	SNR
c	feature correlation
k	number of nonnulls in beta
option	option for split knockoffs

Value

simu_data: a structure contains the following elements

simu_data$fdr_split: a vector recording fdr of split knockoffs w.r.t.nu

simu_data$power_split: a vector recording power of split knockoffs w.r.t.nu

simu_data$fdr_knock: fdr of knockoffs

simu_data$power_knock: power of knockoffs

`simu_unit_cv` *simu_unit_cv*

Description

the simulation unit for simulation experiments with cross validation.

Usage

```
simu_unit_cv (n, p, D_s, A, c, k, option)
```

Arguments

`n`	the sample size
`p`	the dimension of variables
`D_s`	the set of linear transform
`A`	SNR
`c`	feature correlation
`k`	number of nonnulls in beta
`option`	option for split knockoffs

Value

simu_data: a structure contains the following elements

simu_data$fdr: fdr of cv optimal nu in split knockoffs

simu_data$power: power of cv optimal nu in split knockoffs

simu_data$fdr_knock: fdr of knockoffs

simu_data$cv_list: length(D_s) * length(nu_s) matrix with cv loss for each nu for each D

simu_data$chosen_nu: cv selected nu for each D

splitknockoff.decompose *splitknockoff.decompose*

Description

splitknockoff.decompose

Usage

```
splitknockoff.decompose(X, randomize)
```

Arguments

X	the input matrix
randomize	whether to randomize

Value

U

S = S

V = V

U_perp =U_perp

Examples

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

splitknockoff.filter *splitknockoff.filter*

Description

Split Knockoff filter for structural sparsity problem.

Usage

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

Arguments

X	the design matrix
D	the linear transform
y	the response vector
option	options for creating the Knockoff statistics option$eta: the choice of eta for creating the knockoff copy option$q: the desired FDR control bound option$method: 'knockoff' or 'knockoff+' option$stage0: choose the method to conduct split knockoff. 'fixed': fixed intercept assignment for PATH ORDER method. option$beta : the choice of fixed beta for step 0: 'mle': maximum likelihood estimator. 'ridge': ridge regression choice beta with lambda = 1/nu. 'cv_split': cross validation choice of split LASSO over nu and lambda. 'cv_ridge': cross validation choice of ridge regression over lambda. 'path': take the regularization path of split LASSO as the intercept assignment for PATH ORDER method. 'magnitude': using MAGNITUDE method. option$lambda: a set of lambda appointed for path calculation option$nu: a set of nu used for Split Knockoffs option$normalize: whether to normalize the data

Value

results: a cell with the selected variable set in each cell w.r.t. nu.

Z: a cell with the feature significance Z in each cell w.r.t. nu.

t_Z: a cell with the knockoff significance tilde_Z in each cell w.r.t. nu.

Examples

```
k <- 20   # sparsity level
A <- 1    # magnitude
n <- 350  # sample size
p <- 100  # dimension of variables
c <- 0.5  # feature correlation
sigma <-1 # noise level
option <- array(data = NA, dim = length(data), dimnames = NULL)
option$q <- 0.2
option$eta <- 0.1
option$method <- 'knockoff'
option$stage0 <- 'path'
option$normalize <- 'true'
option$cv_rule <- 'min'
option$lambda <- 10.^seq(0, -6, by=-0.01)
option$nu <- 10
option$copy <- 'true'
option <- option[-1]
# generate D
D <- diag(p)
m <- nrow(D)
# generate X
Sigma = matrix(0, p, p)
for( i in 1: p){
  for(j in 1: p){
```

```
    Sigma[i, j] <- c^(abs(i - j))
  }
}
library(mvtnorm)
set.seed(100)
X <- rmvnorm(n,matrix(0, p, 1), Sigma)
# generate beta and gamma
beta_true <- matrix(0, p, 1)
for( i in 1: k){
  beta_true[i, 1] = A
  if ( i%%3 == 1){
    beta_true[i, 1] = -A
  }
}
gamma_true <- D %*% beta_true
S0 <- which(gamma_true!=0)
# generate varepsilon
set.seed(1)
# generate noise and y
varepsilon <- rnorm(n) * sqrt(sigma)
y <- X %*% beta_true + varepsilon
filter_result <- splitknockoff.filter(X, D, y, option)
Z_path <- filter_result$Z
t_Z_path <- filter_result$t_Z
```

`splitknockoff.select` *splitknockoff.select*

Description

splitknockoff.select

Usage

```
splitknockoff.select(W, q)
```

Arguments

`W`	statistics W_j for testing null hypothesis
`q`	target FDR

Value

S array of selected variable indices

split_knockoffs.create

split_knockoffs.create

Description

gives the variable splitting design matrix [A_beta, A_gamma] and response vector tilde_y. It will also create a knockoff copy for A_gamma if required.

Usage

```
split_knockoffs.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$copy true : create a knockoff copy option$eta the choice of eta for creating the 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 [] if option$copy = false.

Examples

```
option <- array(data = NA, dim = length(data), dimnames = NULL)
option$q <- 0.2
option$eta <- 0.1
option$method <- 'knockoff'
option$stage0 <- 'path'
option$normalize <- 'true'
option$cv_rule <- 'min'
option$lambda <- 10.^seq(0, -6, by=-0.01)
option$nu <- 10
option$copy <- 'true'
option <- option[-1]
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  <- split_knockoffs.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
```

`split_knockoffs.statistics.magnitude.cv_mag`
split_knockoffs.statistics.magnitude.cv_mag

Description

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

Usage

```
split_knockoffs.statistics.magnitude.cv_mag(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 calculating cv option$lambda the choice of lambda for the path

Value

lambda: CV optimal lambda

`split_knockoffs.statistics.magnitude.W_mag`
split_knockoffs.statistics.magnitude.W_mag

Description

generate the knockoff statistics W, using the method of magnitude. lambda here is chosen by cross validation.

Usage

```
split_knockoffs.statistics.magnitude.W_mag(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$eta the choice of eta for creating the knockoff copy option$lambda the choice of lambda for the path

Value

W: the knockoff statistics

Z: feature significance

t_Z: knockoff significance

`split_knockoffs.statistics.pathorder.W_fixed`
split_knockoffs.statistics.pathorder.W_fixed

Description

generate the knockoff statistics W for fixed beta in the intercepetion assignment step, using the method of path order.

Usage

```
split_knockoffs.statistics.pathorder.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$eta the choice of eta for creating the knockoff copy option$lambda the choice of lambda for the path option$beta_choice the fixed beta for step 0

Value

W: the knockoff statistics

Z: feature significance

t_Z: knockoff significance

Examples

```
option <- array(data = NA, dim = length(data), dimnames = NULL)
option$q <- 0.2
option$eta <- 0.1
option$method <- 'knockoff'
option$stage0 <- 'path'
option$normalize <- 'true'
option$cv_rule <- 'min'
option$lambda <- 10.^seq(0, -6, by=-0.01)
option$nu <- 10
option$copy <- 'true'
option <- option[-1]
library(mvtnorm)
sigma <-1
n = 350
p = 100
nu = 10
Sigma = matrix(0, p, p)
X <- rmvnorm(n,matrix(0, p, 1), Sigma)
p <- 100
D <- diag(p)
beta_true <- matrix(0, p, 1)
varepsilon <- rnorm(n) * sqrt(sigma)
y <- X %*% beta_true + varepsilon
```

`split_knockoffs.statistics.pathorder.W_path`
split_knockoffs.statistics.pathorder.W_path

Description

generate the knockoff statistics W for beta from a split LASSO path in the intercepetion assignment step, using the method of path order.

Usage

```
split_knockoffs.statistics.pathorder.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$eta the choice of eta for creating the knockoff copy option$lambda the choice of lambda for the path

Value

W: the knockoff statistics

Z: feature significance

t_Z: knockoff significance

Examples

```
k <- 20   # sparsity level
A <- 1    # magnitude
n <- 350  # sample size
p <- 100  # dimension of variables
c <- 0.5  # feature correlation
sigma <-1 # noise level
option <- array(data = NA, dim = length(data), dimnames = NULL)
option$q <- 0.2
option$eta <- 0.1
option$method <- 'knockoff'
option$stage0 <- 'path'
option$normalize <- 'true'
option$cv_rule <- 'min'
option$lambda <- 10.^seq(0, -6, by=-0.01)
option$nu <- 10
option$copy <- 'true'
option <- option[-1]
# generate D
D <- diag(p)
m <- nrow(D)
# generate X
Sigma = matrix(0, p, p)
for( i in 1: p){
  for(j in 1: p){
```

```
    Sigma[i, j] <- c^(abs(i - j))
  }
}
library(mvtnorm)
set.seed(100)
X <- rmvnorm(n,matrix(0, p, 1), Sigma)
# generate beta and gamma
beta_true <- matrix(0, p, 1)
for( i in 1: k){
  beta_true[i, 1] = A
  if ( i%%3 == 1){
    beta_true[i, 1] = -A
  }
}
gamma_true <- D %*% beta_true
S0 <- which(gamma_true!=0)
# generate varepsilon
set.seed(1)
# generate noise and y
varepsilon <- rnorm(n) * sqrt(sigma)
y <- X %*% beta_true + varepsilon
nu = 10
X <- normc(X)
y <- normc(y)
```

Index