Package 'cknockoff'

January 13, 2022

Title The cKnockoff Filter for FDR-controlled Variable Selection

Version 0.0.0.9000

Date 2022-01-13

Description The cKnockoff method is a multiple testing procedure applied to the linear model for variable selection with FDR control. It almost surely dominates the fixed-X knockoff procedure.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

Imports glmnet,
knockoff,
methods,
stats,
utils

Suggests reticulate,
KernSmooth,
doParallel,
parallel,
foreach

R topics documented:

cknockoff *The cKnockoff procedure*

Description

This function applies the cKnockoff procedure to the data (X, y) subjecting to the Gaussian linear model, selecting variables/features relevant for predicting the outcome with FDR control.

Usage

```
cknockoff(
  X,
  y,
  knockoffs = knockoff::create.fixed,
  statistic = stat.glmnet_coefdiff_lm,
  alpha = 0.05,
  Rstar_refine = F,
  n_cores = 1,
  prelim_result = NULL,
  X.pack = NULL
)
```

Arguments

X	n-by-p matrix or data frame of features.
y	response vector of length n.
knockoffs	method used to construct the knockoff matrix for X. It should be a function taking a n-by-p matrix as input and returning a n-by-p matrix of knockoff variables. It is the same as the knockoffs argument in knockoff::knockoff.filter. By default, knockoff::create.fixed is used.
statistic	the knockoff feature statistics (W-statistics) function used to assess variable importance. Any function in the family "statistics" in the R package knockoff that are suitable for the fixed-X setting can be supplied to this argument. But please be aware of the efficiency issue as this function will be called repeatedly in cknockoff. We suggest use the statistic functions provided by our package. By default, a lasso statistic from our package is used.
alpha	nominal false discovery rate (default: 0.05).
Rstar_refine	A logical value determining if we use a better estimation of the number of rejections in calibration. If TRUE, the procedure is cKnockoff* and otherwise is the vanilla cKnockoff. The default is FALSE.
n_cores	the number of cores to be used in computing cKnockoff in parallel. package doParallel is required if n_cores > 1. Otherwise it's computed sequentially.
prelim_result	either a knockoff.result object returned by knockoff::knockoff.filter or a cknockoff.result object returned by cknockoff. cknockoff can read the information from a knockoff result or a cknockoff result and make possibly more rejections with the same FDR control. See more details below.

`X.pack` An object of class `cknockoff.X.pack` returned by function `process_X`. This is used only for simulations studies, where cknockoff is applied many times to the same fixed X, to accelerate computation. General users should ignore it and leave it as default = NULL. If X.pack is provided, the argument "knockoffs" is not needed and will be overwritten.

Details

If argument `prelim_result` is supplied with a object, all the other parameters except `n_cores` and `Rstar_refine` will be overwritten by the information retrieved from it.

If a `knockoff.result` object is supplied to `prelim_result`, cknockoff will return possibly more selections on the same problem with the same FDR control. To use, the function call of `knockoff::knockoff.filter` that returned such `knockoff.result` object must have arguments "statistic" and "fdr" explicitly provided (rather than relying on their defaults). See examples below.

It's possible that cknockoff cannot fetch the function "statistic" or value for "alpha" by their names from a `knockoff.result` object. If this is the case, please supply them to cknockoff explicitly via arguments.

It's possible to even make more rejections based on a `cknockoff.result` object. This is because cknockoff will only explore those promising features and discards the others, say those with large p-values. Calling cknockoff() on a `cknockoff.result` object would further explore some most promising ones among the discarded features and (rarely) possibly make several more rejections. Users can do this recursively and FDR is proved to be controlled whenever they decide to stop. The computational time of each call should be similar.

If the `cknockoff.result` object is obtained by setting `Rstar_refine = FALSE`, more rejections may be made with the same FDR control by supplying `cknockoff.result` to `prelim_result` and set `Rstar_refine = TRUE`. See examples below.

Value

An object of class `cknockoff.result`. It is a list similar to the `knockoff.result` object, containing essentially the same information:

`X`	matrix of original variables (scaled and possibly augmented)
`Xk`	matrix of knockoff variables (cooresponding to the returned X)
`y`	response vector (possibly augmented)
`kn.statistic`	the knockoff feature statistics
`selected`	named vector of selected variables
`sign_predict`	the predicted signs of beta
`record`	a list recording some information during the calculation. It aims to make computing based on "prelim_result" possible and easy. Users may ignore it.

Examples

```
p <- 100; n <- 300; k <- 15
X <- matrix(rnorm(n*p), n)
nonzero <- sample(p, k)
beta <- 2.5 * (1:p %in% nonzero)
```

```
y <- X %*% beta + rnorm(n)
print(which(1:p %in% nonzero))

# Basic usage
result <- cknockoff(X, y, alpha = 0.05, n_cores = 1)
print(result$selected)

# knockoff rejection
library("knockoff")
kn.result <- knockoff.filter(X, y,
                             knockoffs = create.fixed,
                             statistic = stat.glmnet_coefdiff_lm,
                             # must specify this argument explicitly
                             fdr = 0.05
                             # must specify this argument explicitly
                             )
print(kn.result$selected)

# improve knockoff result
result <- cknockoff(prelim_result = kn.result, n_cores = 2)
print(result$selected)

# improve previous cknockoff result
result <- cknockoff(prelim_result = result)
print(result$selected)

# improve previous cknockoff result with cknockoff*
result <- cknockoff(prelim_result = result, n_cores = 2, Rstar_refine = TRUE)
print(result$selected)
```

`create.fixed.MRC` *Creates fixed-X knockoff variables by MRC*

Description

This function is a R wrapper of the knocproduce_FX_knockoffskpy function in the Python package knockpy. It creates fixed-X knockoff variables by Minimizing Reconstructability (see https://arxiv.org/abs/2011.14625). To use this function, Users should have the R package `reticulate` and the python package knockpy installed.

Usage

```
create.fixed.MRC(
  X,
  method = c("mvr", "maxent", "mmi", "sdp", "equicorrelated", "ci"),
  solver = NULL,
  knockpy = NULL,
  num_processes = 1
)
```

Arguments

X	n-by-p matrix of the features.
method	Same argument as the "method" in the smatrix.compute_smatrix function in knockpy package. Take value from "mvr", "maxent", "mmi", "sdp", "equicorrelated", and "ci". The default is "mvr".
solver	Same argument as the "solver" in the smatrix.compute_smatrix function in knockpy package. Take value from "cd", "sdp", "psgd". Can be leave as NULL.
knockpy	The R object wrapping the Python module "knockpy". Users may provide their own wrapper using "reticulate::import" if "knockpy" is not installed in the default python environment (e.g. in some environments created by anaconda).
num_processes	Same argument as the "num_processes" in the smatrix.compute_smatrix function in knockpy package. Number of parallel process to use in computing the matrix approximately.

Value

A list containing the following components:

X	n-by-p matrix of original variables (normalized).
Xk	n-by-p matrix of knockoff variables.

---

process_X *Compute and store matrices needed by cknockoff() from on X*

Description

Compute and store matrices needed by cknockoff() from on X

Usage

```
process_X(X, knockoffs = knockoff::create.fixed)
```

Arguments

X	X n-by-p matrix of original variables.
knockoffs	either knockoff matrix of X or a knockoffs function that can generate it. If the knockoff matrix is supplied, both X and knockoffs should be properly normalized, e.g. using the returned X and Xk of the knockoff::create.fixed function. By default, knockoff::create.fixed is used.

Value

An object of class "cknockoff.X.pack". This object is a list containing many matrices like the knockoff matrix and a basis of the linear space spanned by X, etc. Users don't need to work on the object themselves but pass it to the "X.pack" parameter in cknockoff() if needed.

Examples

```
p <- 100; n <- 300; k <- 15
X <- matrix(rnorm(n*p), n)
nonzero <- sample(p, k)
beta <- 2.5 * (1:p %in% nonzero)
y <- X %*% beta + rnorm(n)
print(which(1:p %in% nonzero))

X.pack <- process_X(X, knockoffs = knockoff::create.fixed)

result <- cknockoff(X, y,
                    alpha = 0.05,
                    n_cores = 1,
                    X.pack = X.pack)
print(result$selected)
```

`stat.glmnet_coefdiff_lm` *Lasso Coefficient-Difference feature statistics with tiebreaker*

Description

Fit Lasso on the augmented model $y\ [X, \widetilde{X}]$ with regularization parameter λ. Then, compute the difference statistic

$$W_j^0 = |Z_j| - |\tilde{Z}_j|$$

where Z_j and $\tilde{Z}_j$ are the coefficient estimates for the jth variable and its knockoff, respectively. For those variables that both themselves and their knockoffs are not selected by Lasso, we break the tie by their correlation with the residue, by defining

$$\rho_j = |X_j^T residue| - |\tilde{X}_j^T residue|$$

for them and $\rho_j = 2\lambda$ for the others. The final feature statistics are

$$W_j = W_j^0 + \rho_j.$$

λ is set to be $2\tilde{\sigma}$, where $\tilde{\sigma}$ is an estimator of the noise level that is independent of $[X, \widetilde{X}]^T y$.

Usage

```
stat.glmnet_coefdiff_lm(X, X_k, y, sigma_tilde = NULL)
```

Arguments

X	n-by-p matrix of original variables.
X_k	n-by-p matrix of knockoff variables.
y	y vector of length n, containing the response variables.
sigma_tilde	An estimator of the noise level that is independent of $[X, \widetilde{X}]^T y$. If not provided, it will be computed inside the function based on X, X_k, and y, in which case we must have $n > 2p$.

Details

If sigma_tilde is not provided and $n = 2p$, sigma_tilde will be computed from the residue of the OLS fitting y X. The resulting estimator is thus not independent of $[X, \widetilde{X}]^T y$. Users should avoid this case if they want a guaranteed FDR control. Though in practice it shouldn't really make a difference.

Details of the calculation of this feature statistics can be found in (our paper).

The implementation of this function is modified from the `knockoff::stat.glmnet_coefdiff()` function.

Value

A vector of knockoff feature statistics W of length p.

See Also

Other statistics: `stat.glmnet_lambdasmax_lm()`

Examples

```
p <- 100; n <- 300; k <- 15
X <- matrix(rnorm(n*p), n)
nonzero <- sample(p, k)
beta <- 2.5 * (1:p %in% nonzero)
y <- X %*% beta + rnorm(n)
print(which(1:p %in% nonzero))

result <- cknockoff(X, y,
                    knockoffs = knockoff::create.fixed,
                    statistic = stat.glmnet_coefdiff_lm,
                    alpha = 0.05, n_cores = 1)
print(result$selected)
```

`stat.glmnet_lambdasmax_lm`

Efficient Lasso Signed-Max feature statistics

Description

This function provides a computationally efficient feature statistic that behave similarly to the Lasso Signed-Max feature statistics produced by the `knockoff::stat.glmnet_lambdasmax()` function. We compute a $\hat{\lambda}_j$ as an approximation of the maximal regularization parameter λ at which the jth variable enters the Lasso model. And obtain the feature statistics

$$W_j = (\hat{\lambda}_j \vee \hat{\lambda}_{j+m}) \cdot sgn(\hat{\lambda}_j - \hat{\lambda}_{j+m})$$

Usage

```
stat.glmnet_lambdasmax_lm(X, X_k, y, sigma_tilde = NULL, nlambda = 10)
```

Arguments

`X`	n-by-p matrix of original variables.
`X_k`	n-by-p matrix of knockoff variables.
`y`	y vector of length n, containing the response variables.
`sigma_tilde`	An estimator of the noise level that is independent of $[X, \widetilde{X}]^T y$. If not provided, it will be computed inside the function based on X, X_k, and y, in which case we must have $n > 2p$.
`nlambda`	the number of grid points in computing the Lasso path. A larger value will make the calculation more precise but more expensive. The default value is 10.

Details

If sigma_tilde is not provided and $n = 2p$, sigma_tilde will be computed from the residue of the OLS fitting y X. The resulting estimator is thus not independent of $[X, \widetilde{X}]^T y$. Users should avoid this case if they want a guaranteed FDR control. Though in practice it shouldn't really make a difference.

Details of the calculation of this feature statistics can be found in (our paper).

The implementation of this function is modified from the `knockoff::stat.glmnet_lambdasmax()` function.

Value

A vector of knockoff feature statistics W of length p.

See Also

Other statistics: `stat.glmnet_coefdiff_lm()`

Examples

```
p <- 100; n <- 300; k <- 15
X <- matrix(rnorm(n*p), n)
nonzero <- sample(p, k)
beta <- 2.5 * (1:p %in% nonzero)
y <- X %*% beta + rnorm(n)
print(which(1:p %in% nonzero))

result <- cknockoff(X, y,
                    knockoffs = knockoff::create.fixed,
                    statistic = stat.glmnet_lambdasmax_lm,
                    alpha = 0.05, n_cores = 1)
print(result$selected)
```

Index