Package ‘kko’

February 1, 2022

Title  Kernel Knockoffs Selection for Nonparametric Additive Models

Version  1.0.1


License  GPL (>= 2)

Depends  R (>= 3.6.3)

Imports  grpreg, knockoff, doParallel, parallel, foreach, ExtDist

Suggests  knitr, rmarkdown, ggplot2

Encoding  UTF-8

LazyData  false

RoxygenNote  7.1.2

VignetteBuilder  knitr

Author  Xiaowu Dai [aut],
        Xiang Lyu [aut, cre],
        Lexin Li [aut]

Maintainer  Xiang Lyu <xianglyu@berkeley.edu>

NeedsCompilation  no

Repository  CRAN

Date/Publication  2022-02-01 09:10:05 UTC

R topics documented:

  generate_data .................................................. 2
  kko ............................................................. 3
  KO_evaluation .................................................. 4
  rk_fit .......................................................... 5
  rk_subsample .................................................. 7
  rk_tune ......................................................... 8
Description

The function generate response from additive models of various components.

Usage

generate_data(X, reg_coef, model = "linear", err_sd = 1)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>design matrix of additive model; rows are observations and columns are variables.</td>
</tr>
<tr>
<td>reg_coef</td>
<td>regression coefficient vector.</td>
</tr>
<tr>
<td>model</td>
<td>types of components. Default is &quot;linear&quot;. Other choices are</td>
</tr>
<tr>
<td></td>
<td>linear regression.</td>
</tr>
<tr>
<td></td>
<td>poly polynomial of degree sampled from 2 to 4.</td>
</tr>
<tr>
<td></td>
<td>sinpoly sum of polynomial of sin and cos.</td>
</tr>
<tr>
<td></td>
<td>sinratio ratio of sin.</td>
</tr>
<tr>
<td></td>
<td>sinmix sampled from poly and sinratio.</td>
</tr>
<tr>
<td>err_sd</td>
<td>standard deviation of regression error.</td>
</tr>
</tbody>
</table>

Value

response vector

Author(s)

Xiaowu Dai, Xiang Lyu, Lexin Li

Examples

```r
p=5 # number of predictors
s=2 # sparsity, number of nonzero component functions
sig_mag=100 # signal strength
n= 200 # sample size
model="poly" # component function type
X=matrix(rnorm(n*p),n,p) %*% chol(toeplitz(0.3^(0:(p-1)))) # generate design
reg_coef=c(rep(1,s),rep(0,p-s)) # regression coefficient
reg_coef=reg_coef*(2*(rnorm(p)>0)-1)*sig_mag
y=generate_data(X,reg_coef,model) # response vector
```
variable selection for additive model via KKO

Description

The function applys KKO to compute importance scores of components.

Usage

```r
kko(
  X,
  y,
  X_k,
  rfn_range = c(2, 3, 4),
  n_stb_tune = 50,
  n_stb = 100,
  cv_folds = 10,
  frac_stb = 1/2,
  nCores_para = 4,
  rkernel = c("laplacian", "gaussian", "cauchy"),
  rk_scale = 1
)
```

Arguments

- `X`: design matrix of additive model; rows are observations and columns are variables.
- `y`: response of additive model.
- `X_k`: knockoffs matrix of design; the same size as `X`.
- `rfn_range`: a vector of random feature expansion numbers to be tuned.
- `n_stb_tune`: number of subsampling for tuning random feature numbers.
- `n_stb`: number of subsampling for computing importance scores.
- `cv_folds`: the folds of cross-validation for tuning group lasso penalty.
- `frac_stb`: fraction of subsample size.
- `nCores_para`: number of cores for parallelizing subsampling.
- `rkernel`: kernel choices. Default is "laplacian". Other choices are "cauchy" and "gaussian".
- `rk_scale`: scale parameter of sampling distribution for random feature expansion. For gaussian kernel, it is standard deviation of gaussian sampling distribution.

Value

a list of selection results.
importance_score: importance scores of variables for knockoff filtering.

selection_frequency: a 0/1 matrix of selection results on subsamples. Rows are subsamples, and columns are variables. The first half columns are variables of design X, and the latter are knockoffs X_k.

rfn_tune: tuned optimal random feature number.

rfn_range: range of random feature numbers.

tune_result: a list of tuning results.

Author(s)

Xiaowu Dai, Xiang Lyu, Lexin Li

Examples

```r
library(knockoff)
p=4 # number of predictors
sig_mag=100 # signal strength
n= 100 # sample size
rkernel="laplacian" # kernel choice
s=2 # sparsity, number of nonzero component functions
rk_scale=1 # scaling parameter of kernel
rpn_range=c(2,3,4) # number of random features
cv_folds=15 # folds of cross-validation in group lasso
n_stb=10 # number of subsampling for importance scores
n_stb_tune=5 # number of subsampling for tuning random feature number
frac_stb=1/2 # fraction of subsample
nCores_para=2 # number of cores for parallelization
X=matrix(rnorm(n*p),n,p)%*%chol(toeplitz(0.3^((0:(p-1)))))) # generate design
X_k = create.second.order(X) # generate knockoff
reg_coef=c(rep(1,s),rep(0,p-s)) # regression coefficient
reg_coef=reg_coef*(2*(rnorm(p)>0)-1)*sig_mag
y=X%*%reg_coef + rnorm(n) # response
kko(X,y,X_k,rfn_range,n_stb_tune,n_stb,cv_folds,frac_stb,nCores_para,rkernel,rk_scale)
```

KO_evaluation: evaluate performance of KKO selection

Description

The function computes \{FDP, FPR, TPR\} of selection by knockoff filtering on importance scores of KKO.

Usage

```r
KO_evaluation(W, reg_coef, fdr_range = 0.2, offset = 1)
```
rk_fit

Arguments

- \( W \): importance scores of variables.
- \( \text{reg}_\text{coef} \): true regression coefficient.
- \( \text{fdr}_\text{range} \): FDR control levels of knockoff filter.
- \( \text{offset} \): 0/1. If 1, knockoff+ filter. Otherwise, knockoff filter.

Value

FDP, FPR, TPR of knockoff filtering at \( \text{fdr}_\text{range} \).

Author(s)

Xiaowu Dai, Xiang Lyu, Lexin Li

Examples

```r
library(knockoff)
p=5 # number of predictors
sig_mag=100 # signal strength
n=100 # sample size
rkernel="laplacian" # kernel choice
s=2 # sparsity, number of nonzero component functions
rk_scale=1 # scaling parameter of kernel
rfn_range=c(2,3,4) # number of random features
cv_folds=15 # folds of cross-validation in group lasso
n_stb=10 # number of subsampling for importance scores
n_stb_tune=5 # number of subsampling for tuning random feature number
frac_stb=1/2 # fraction of subsample
nCores_para=2 # number of cores for parallelization
X=matrix(rnorm(n*p),n,p)%*%chol(toeplitz(0.3^(0:(p-1)))) # generate design
X_k = create.second_order(X) # generate knockoff
reg_coef=c(rep(1,s),rep(0,p-s)) # regression coefficient
reg_coef=reg_coef*(2*(rnorm(p)>0)-1)*sig_mag
y=X%*% reg_coef + rnorm(n) # response
kko_fit=kko(X,y,X_k,rfn_range,n_stb_tune,n_stb,cv_folds,frac_stb,nCores_para,rkernel,rk_scale)
W=kko_fit$importance_score
fdr_range=c(0.2,0.3,0.4,0.5)
KO_evaluation(W,reg_coef,fdr_range,offset=1)
```

rk_fit nonparametric additive model selection via random kernel

Description

The function selects additive components via applying group lasso on random feature expansion of data and knockoffs.
Usage

rk_fit(
  X,
  y,
  X_k,
  rfn,
  cv_folds,
  rkernel = "laplacian",
  rk_scale = 1,
  rseed = NULL
)

Arguments

X  design matrix of additive model; rows are observations and columns are variables.
y  response of additive model.
X_k knockoffs matrix of design; the same size as X.
rfn random feature expansion number.
cv_folds the folds of cross-validation for tuning group lasso penalty.
rkernel kernel choices. Default is "laplacian". Other choices are "cauchy" and "gaussian".
rk_scale scaling parameter of sampling distribution for random feature expansion. For gaussian kernel, it is standard deviation of gaussian sampling distribution.
rseed seed for random feature expansion.

Value

a 0/1 vector indicating selected components.

Author(s)

Xiaowu Dai, Xiang Lyu, Lexin Li

Examples

library(knockoff)
p=5 # number of predictors
sig_mag=100 # signal strength
n= 200 # sample size
rkernel="laplacian" # kernel choice
s=2 # sparsity, number of nonzero component functions
rk_scale=1 # scaling parameter of kernel
rfn= 3 # number of random features
cv_folds=15 # folds of cross-validation in group lasso
X=matrix(rnorm(n*p),n,p)%*%chol(toeplitz(0.3^(0:(p-1)))) # generate design
X_k = create.second_order(X) # generate knockoff
reg_coef=c(rep(1,s),rep(0,p-s)) # regression coefficient
reg_coef = reg_coef * (2 * (rnorm(p) > 0) - 1) * sig_mag
y = X %*% reg_coef + rnorm(n) # response

# the first half is variables of design X, and the latter is knockoffs X_k
rk_fit(X, y, X_k, rfn, cv_folds, rkernel, rk_scale)

---

rk_subsample

**compute selection frequency of rk_fit on subsamples**

**Description**

The function applys rk_fit on subsamples and record selection results.

**Usage**

```r
rk_subsample(
  X,
  y,
  X_k,
  rfn,
  n_stb,
  cv_folds,
  frac_stb = 1/2,
  nCores_para,
  rkernel = "laplacian",
  rk_scale = 1
)
```

**Arguments**

- **X**: design matrix of additive model; rows are observations and columns are variables.
- **y**: response of additive model.
- **X_k**: knockoffs matrix of design; the same size as X.
- **rfn**: random feature expansion number.
- **n_stb**: number of subsampling.
- **cv_folds**: the folds of cross-validation for tuning group lasso.
- **frac_stb**: fraction of subsample size.
- **nCores_para**: number of cores for parallelizing subsampling.
- **rkernel**: kernel choices. Default is "laplacian". Other choices are "cauchy" and "gaussian".
- **rk_scale**: scaling parameter of sampling distribution for random feature expansion. For gaussian kernel, it is standard deviation of gaussian sampling distribution.
Value

a 0/1 matrix indicating selection results. Rows are subsamples, and columns are variables. The first half columns are variables of design $X$, and the latter are knockoffs $X_k$.

Author(s)

Xiaowu Dai, Xiang Lyu, Lexin Li

Examples

```r
library(knockoff)
p=5 # number of predictors
sig_mag=100 # signal strength
n= 100 # sample size
rkernel="laplacian" # kernel choice
s=2 # sparsity, number of nonzero component functions
rk_scale=1 # scaling parameter of kernel
rfn= 3 # number of random features
cv_folds=15 # folds of cross-validation in group lasso
n_stb=10 # number of subsampling
frac_stb=1/2 # fraction of subsamples
nCores_para=2 # number of cores for parallelization
X=matrix(rnorm(n*p),n,p)%*%chol(toeplitz(0.3^(0:(p-1)))) # generate design
X_k = create.second_order(X) # generate knockoff
reg_coef=c(rep(1,s),rep(0,p-s)) # regression coefficient
reg_coef=reg_coef*(2*(rnorm(p)>0)-1)*sig_mag
y=X%*% reg_coef + rnorm(n) # response
rk_subsample(X,y,X_k,rfn,n_stb,cv_folds,frac_stb,nCores_para,rkernel,rk_scale)
```

---

rk_tune

_tune random feature number for KKO._

Description

The function applies KKO with different random feature numbers to tune the optimal number.

Usage

```r
rk_tune(
  X, 
  y, 
  X_k, 
  rfn_range, 
  n_stb, 
  cv_folds, 
  frac_stb = 1/2,
)```
nCores_para = 1,
rkernel = "laplacian",
rk_scale = 1
)

Arguments

X design matrix of additive model; rows are observations and columns are variables.
y response of additive model.
X_k knockoffs matrix of design; the same size as X.
rfn_range a vector of random feature expansion numbers to be tuned.
n_stb number of subsampling in KKO.
cv_folds the folds of cross-validation for tuning group lasso.
frac_stb fraction of subsample.
nCores_para number of cores for parallelizing subsampling.
rkernel kernel choices. Default is "laplacian". Other choices are "cauchy" and "gaussian".
rk_scale scaling parameter of sampling distribution for random feature expansion. For gaussian kernel, it is standard deviation of gaussian sampling distribution.

Value

a list of tuning results.

rfn_tune tuned optimal random feature number.
rfn_range a vector of random feature expansion numbers to be tuned.
scores scores of random feature numbers. rfn_tune has the maximal score.
Pi_list a list of subsample selection results for each random feature number.

Author(s)

Xiaowu Dai, Xiang Lyu, Lexin Li

Examples

library(knockoff)
p=5 # number of predictors
sig_mag=100 # signal strength
n= 100 # sample size
rkernel="laplacian" # kernel choice
s=2 # sparsity, number of nonzero component functions
rk_scale=1 # scaling parametere of kernel
rfn_range= c(2,3,4) # number of random features
cv_folds=15 # folds of cross-validation in group lasso
n_stb=10 # number of subsampling
frac_stb=1/2 # fraction of subsample
nCores_para=2 # number of cores for parallelization
X=matrix(rnorm(n*p),n,p)%*%chol(toeplitz(0.3^(0:(p-1)))) # generate design
X_k = create.second_order(X) # generate knockoff
reg_coef=c(rep(1,s),rep(0,p-s)) # regression coefficient
reg_coef=reg_coef*(2*(rnorm(p)>0)-1)*sig_mag
y=X%*%reg_coef + rnorm(n) # response

rk_tune(X,y,X_k,rfn_range,n_stb,cv_folds,frac_stb,nCores_para,rkernel,rk_scale)
Index

generate_data, 2

kko, 3
KO_evaluation, 4

rk_fit, 5
rk_subsample, 7
rk_tune, 8