Package ‘EAinference’

October 12, 2022

Type Package

Title Estimator Augmentation and Simulation-Based Inference

Version 0.2.3

Maintainer Seunghyun Min <seunghyun@ucla.edu>

Description Estimator augmentation methods for statistical inference on high-dimensional data, as described in Zhou, Q. (2014) <arXiv:1401.4425v2> and Zhou, Q. and Min, S. (2017) <doi:10.1214/17-EJS1309>. It provides several simulation-based inference methods: (a) Gaussian and wild multiplier bootstrap for lasso, group lasso, scaled lasso, scaled group lasso and their de-biased estimators, (b) importance sampler for approximating p-values in these methods, (c) Markov chain Monte Carlo lasso sampler with applications in post-selection inference.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Depends R (>= 3.2.3)

Imports stats, graphics, msm, mvtnorm, parallel, limSolve, MASS, hdi, Rcpp

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 6.0.1

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

NeedsCompilation yes

Author Seunghyun Min [aut, cre], Qing Zhou [aut]

Repository CRAN

Date/Publication 2017-12-02 00:01:31 UTC
cv.lasso

R topics documented:

- cv.lasso ................................................................. 2
- hdIS ................................................................. 3
- lassoFit ............................................................. 5
- MHLS ................................................................. 7
- PB.CI ................................................................. 9
- PBsampler .......................................................... 10
- plot.MHLS ......................................................... 12
- postInference.MHLS ............................................. 13
- print.MHLS ......................................................... 15
- summary.MHLS .................................................... 16

Index 17

---

cv.lasso

*Compute K-fold cross-validated mean squared error for lasso*

**Description**

Computes K-fold cross-validated mean squared error to propose a lambda value for lasso, group lasso, scaled lasso or scaled group lasso.

**Usage**

```r
cv.lasso(X, Y, group = 1:ncol(X), weights = rep(1, max(group)), type, K = 10L, minlbd, maxlbd, num.lbdseq = 100L, parallel = FALSE, ncores = 2L, plot.it = FALSE, verbose = FALSE)
```

**Arguments**

- `X` predictor matrix.
- `Y` response vector.
- `group` p x 1 vector of consecutive integers describing the group structure. The number of groups should be the same as `max(group)`. Default is `group = 1:p`, where `p` is number of covariates. See examples for a guideline.
- `weights` weight vector with length equal to the number of groups. Default is `rep(1, max(group))`.
- `type` type of penalty. Must be specified to be one of the following: "lasso", "grlasso", "slasso" or "sgrlasso", which correspond to lasso, group lasso, scaled lasso or scaled group lasso.
- `K` integer. Number of folds
- `minlbd` numeric. Minimum value of the lambda sequence.
- `maxlbd` numeric. Maximum value of the lambda sequence.
- `num.lbdseq` integer. Length of the lambda sequence.
hdIS

**parallel**  logical. If parallel = TRUE, uses parallelization. Default is parallel = FALSE.

**ncores**  integer. The number of cores to use for parallelization.

**plot.it**  logical. If true, plots the squared error curve.

**verbose**  logical.

**Value**

- **lbd.min**  a value of lambda which gives a minimum squared error.
- **lbd.1se**  a largest lambda within 1 standard error from lbd.min.
- **lbd.seq**  lambda sequence.
- **cv**  mean squared error at each lambda value.
- **cvsd**  the standard deviation of cv.

**Examples**

```r
set.seed(123)
n <- 30
p <- 50
group <- rep(1:(p/10),each=10)
weights <- rep(1, max(group))
X <- matrix(rnorm(n*p),n)
truebeta <- c(rep(1,5),rep(0,p-5))
Y <- X%*%truebeta + rnorm(n)

# To accelerate the computational time, we set K=2 and num.lbdseq=2.
# However, in practice, Allowing K=10 and num.lbdseq > 100 is recommended.
cv.lasso(X = X, Y = Y, group = group, weights = weights, K = 2,
type = "grlasso", num.lbdseq = 2, plot.it = FALSE)
cv.lasso(X = X, Y = Y, group = group, weights = weights, K = 2,
type = "sgrlasso", num.lbdseq = 2, plot.it = FALSE)
```

**Description**

hdIS computes importance weights using samples drawn by PBsampler. See the examples below for details.

**Usage**

hdIS(PBsample, PETarget, sig2Target, lbdTarget, TsA.method = "default",
log = TRUE, parallel = FALSE, ncores = 2L)
Arguments

PBsample, PETarget, sig2Target, lbdTarget

Parameters of target distribution. (point estimate of beta or E(y), estimated variance of error and lambda)

TsA.method

Method to construct T(eta(s), A) matrix. See Zhou and Min(2017) for details.

log

Logical. If log = TRUE, importance weight is computed in log scale.

parallel

Logical. If parallel = TRUE, uses parallelization. Default is parallel = FALSE.

ncores

Integer. The number of cores to use for parallelization.

Details

Computes importance weights which is defined as (target density)/(proposal density), when the samples are drawn from the proposal distribution with the function PBsampler while the parameters of the target distribution are (PETarget, sig2Target, lbdTarget).

Say that we are interested in computing the expectation of a function of a random variable, h(X).

Let f(x) be the true or target distribution and g(x) be the proposal distribution. We can approximate the expectation, E[h(X)], by a weighted average of samples, x_i, drawn from the proposal distribution as follows, E[h(X)] = mean( h(x_i) * f(x_i)/h(x_i) ).

Value

Importance weights of the proposed samples.

References


Examples

set.seed(1234)
n <- 10
p <- 30
Niter <- 10
Group <- rep(1:(p/10), each = 10)
Weights <- rep(1, p/10)
x <- matrix(rnorm(n*p), n)

# Target distribution parameter
PETarget <- rep(0, p)
sig2Target <- .5
lbdTarget <- .37

# Using non-mixture distribution
lassoFit

# ------------------------------
## Proposal distribution parameter
PEProp1 <- rep(1, p)
sig2Prop1 <- .5
lbdProp1 <- 1
PB <- PBsampler(X = x, PE_1 = PEProp1, sig2_1 = sig2Prop1,
lbd_1 = lbdProp1, weights = Weights, group = Group, niter = Niter,
type="grlasso", PEtype = "coeff")

hdIS(PB, PETarget = PETarget, sig2Target = sig2Target, lbdTarget = lbdTarget,
log = TRUE)

# Using mixture distribution
# ------------------------------
# Target distribution parameters (coeff, sig2, lbd) = (rep(0,p), .5, .37)
# Proposal distribution parameters
# (coeff, sig2, lbd) = (rep(0,p), .5, .37) & (rep(1,p), 1, .5)

PEProp1 <- rep(0,p); PEProp2 <- rep(1,p)
sig2Prop1 <- .5; sig2Prop2 <- 1
lbdProp1 <- .37; lbdProp2 <- .5

PBMixture <- PBsampler(X = x, PE_1 = PEProp1,
sig2_1 = sig2Prop1, lbd_1 = lbdProp1, PE_2 = PEProp2,
sig2_2 = sig2Prop2, lbd_2 = lbdProp2, weights = Weights, group = Group,
niter = Niter, type = "grlasso", PEtype = "coeff")
hdIS(PBMixture, PETarget = PETarget, sig2Target = sig2Target, lbdTarget = lbdTarget,
log = TRUE)

---

lassoFit  Compute lasso estimator

Description
Computes lasso, group lasso, scaled lasso, or scaled group lasso solution. The outputs are coefficient-estimate and subgradient. If type = "slasso" or type = "sgrlasso", the output will include estimated standard deviation.

Usage
lassoFit(X, Y, type, lbd, group = 1:ncol(X), weights = rep(1, max(group)),
verbose = FALSE, ...)

Arguments
X predictor matrix.
Y response vector.
lassoFit

Parameters:

- **type**: type of penalty. Must be specified to be one of the following: "lasso", "grlasso", "slasso" or "sgrlasso", which correspond to lasso, group lasso, scaled lasso or scaled group lasso.

- **lbd**: penalty term of lasso. By letting this argument be "cv.1se" or "cv.min", users can have the cross-validated lambda that gives either minimum squared error or that is within 1 std error bound.

- **group**: p x 1 vector of consecutive integers describing the group structure. The number of groups should be the same as max(group). Default is group = 1:p, where p is number of covariates.

- **weights**: weight vector with length equal to the number of groups. Default is weights = rep(1, max(group)).

- **verbose**: logical. Only available for type = "slasso" or type = "sgrlasso".

- **...**: auxiliary arguments for lbd = "cv.min", lbd = "cv.1se". See cv.lasso for details.

Details

Computes lasso, group lasso, scaled lasso, or scaled group lasso solution. Users can specify the value of lbd or choose to run cross-validation to get optimal lambda in term of mean squared error. Coordinate decent algorithm is used to fit scaled lasso and scaled group lasso models.

Value

- **B0**: coefficient estimator.
- **S0**: subgradient.
- **sigmaHat**: estimated standard deviation.
- **lbd**, **weights**, **group**: same as input arguments.

References


Examples

```r
set.seed(123)
n <- 50
p <- 10
X <- matrix(rnorm(n*p), n)
Y <- X %*% c(1, 1, rep(0, p-2)) + rnorm(n)
# lasso
# lassoFit(X = X, Y = Y, type = "lasso", lbd = .5)
```
MHLS

Metropolis-Hastings lasso sampler under a fixed active set.

Description

Metropolis-Hastings sampler to simulate from the sampling distribution of lasso given a fixed active set.

Usage

MHLS(X, PE, sig2, lbd, weights = rep(1, ncol(X)), B0, S0, A = which(B0 != 0), tau = rep(1, ncol(X)), niter = 2000, burnin = 0, PEtype = "coeff", updateS.itv = 1, verbose = FALSE, ...)

Arguments

X predictor matrix.
PE, sig2, lbd parameters of target distribution. (point estimate of beta or E(y) depends on PEtype, variance estimate of error and lambda).
weights weight vector with length p(the number of covariates). Default is weights = rep(1, p).
B0 numeric vector with length p. Initial value of lasso estimator.
S0 numeric vector with length p. Initial value of subgradients. If not given, this will be generated in a default way.
A numeric vector. Active coefficient index. Every active coefficient index in B0 must be included. Default is A = which(B0 != 0).
tau numeric vector with length p. Standard deviation of proposal distribution for each coefficient.
niter integer. The number of iterations. Default is niter = 2000
burnin integer. The length of burin-in periods. Default is burnin = 0
PEtype  
Type of PE which is needed to characterize the target distribution. Users can choose either "coeff" or "mu".

updateS.itv  
integer. Update subgradients every updateS.itv iterations. Set this value larger than niter if one wants to skip updating subgradients.

verbose  
logical. If true, print out the progress step.

Details  
Given appropriate initial value, provides Metropolis-Hastings samples under the fixed active set. From the initial values, B0 and S0, MHLS draws beta and subgrad samples. In every iteration, given t-th iteration values, t-th beta and t-th subgrad, a new set of proposed beta and subgradient is sampled. We either accept the proposed sample and use that as (t+1)-th iteration values or reuse t-th iteration values. See Zhou(2014) for more details.

Value  
MHLS returns an object of class "MHLS". The functions summary.MHLS and plot.MHLS provide a brief summary and generate plots.

beta  
lasso samples.

subgrad  
subgradient samples.

acceptHistory  
numbers of acceptance and proposal.

Examples  
#-------------------------
# Low dim
#-------------------------
set.seed(123)
n <- 10
p <- 5
PB.CI

Provide (1-alpha)% confidence interval of each coefficients

Description

Using samples drawn by PBsampler, computes (1-alpha)% confidence interval of each coefficient.

Usage

PB.CI(object, alpha = 0.05, method = "debias", parallel = FALSE, ncores = 2L)

Arguments

object bootstrap samples of class PB from PBsampler
alpha significance level.
PBsampler

Parametric bootstrap sampler for lasso, group lasso, scaled lasso or scaled group lasso estimator

Description

Draw gaussian bootstrap or wild multiplier bootstrap samples for lasso, group lasso, scaled lasso and scaled group lasso estimators along with their subgradients.

Usage

PBsampler(X, PE_1, sig2_1, lbd_1, PE_2, sig2_2, lbd_2, weights = rep(1, max(group)), group = 1:ncol(X), niter = 2000, type, PEtype = "coeff", Btype = "gaussian", Y = NULL, parallel = FALSE, ncores = 2L, verbose = FALSE)
**Arguments**

- **X**
  - predictor matrix.

- **PE_1, sig2_1, lbd_1**
  - parameters of target distribution. (point estimate of beta or $E(y)$ depends on PEtype, variance estimate of error and lambda) sig2_1 is only needed when Btype = "wild".

- **PE_2, sig2_2, lbd_2**
  - additional parameters of target distribution. This is required only if mixture distribution is used. sig2_2 is only needed when Btype = "wild".

- **weights**
  - weight vector with length equal to the number of groups. Default is `rep(1, max(group))`.

- **group**
  - p x 1 vector of consecutive integers describing the group structure. The number of groups should be the same as max(group). Default is `group = 1:p`, where p is number of covariates. See examples for a guideline.

- **niter**
  - integer. The number of iterations. Default is `niter = 2000`.

- **type**
  - type of penalty. Must be specified to be one of the following: "lasso", "grlasso", "slasso" or "sgrlasso".

- **PEtype**
  - Type of PE which is needed to characterize the target distribution. Users can choose either "coeff" or "mu".

- **Btype**
  - Type of bootstrap method. Users can choose either "gaussian" for gaussian bootstrap or "wild" for wild multiplier bootstrap. Default is "gaussian".

- **Y**
  - response vector. This is only required when Btype = "wild".

- **parallel**
  - logical. If parallel = TRUE, uses parallelization. Default is parallel = FALSE.

- **ncores**
  - integer. The number of cores to use for parallelization.

- **verbose**
  - logical. This works only when parallel = FALSE.

**Details**

This function provides bootstrap samples for lasso, group lasso, scaled lasso or scaled group lasso estimator and its subgradient.

The sampling distribution is characterized by (PE, sig2, lbd). If Btype = "gaussian", error_new is generated from $N(0, \text{sig2})$. If Btype = "wild", we first generate error_new from $N(0, 1)$ and multiply with the residuals. Then, if PEtype = "coeff", y_new is generated by $X * \text{PE} + \text{error_new}$ and if PEtype = "mu", y_new is generated by $\text{PE} + \text{error_new}$.

By providing (PE_2, sig2_2, lbd_2), this function simulates from a mixture distribution. With 1/2 probability, samples will be drawn from the distribution with parameters (PE_1, sig2_1, lbd_1) and with another 1/2 probability, they will be drawn from the distribution with parameters (PE_2, sig2_2, lbd_2). Four distinct penalties can be used; "lasso" for lasso, "grlasso" for group lasso, "slasso" for scaled lasso and "sgrlasso" for scaled group lasso. See Zhou(2014) and Zhou and Min(2017) for details.

**Value**

- **beta**
  - coefficient estimate.

- **subgrad**
  - subgradient.
hsigma  standard deviation estimator, for type="lasso" or type="sgrlasso" only.
X, PE, sig2, weights, group, type, PEtype, Btype, Y, mixture
  model parameters.

References


Examples

set.seed(1234)
n <- 10
p <- 30
Niter <- 10
Group <- rep(1:(p/10), each = 10)
Weights <- rep(1, p/10)
x <- matrix(rnorm(n*p), n)
#
# Using non-mixture distribution
#
PBSampler(X = x, PE_1 = rep(0, p), sig2_1 = 1, lbd_1 = .5,
weights = Weights, group = Group, type = "grlasso", niter = Niter, parallel = FALSE)
PBSampler(X = x, PE_1 = rep(0, p), sig2_1 = 1, lbd_1 = .5,
weights = Weights, group = Group, type = "grlasso", niter = Niter, parallel = TRUE)
#
# Using mixture distribution
#
PBSampler(X = x, PE_1 = rep(0, p), sig2_1 = 1, lbd_1 = .5,
PE_2 = rep(1, p), sig2_2 = 2, lbd_2 = .3, weights = Weights,
group = Group, type = "grlasso", niter = Niter, parallel = TRUE)

Description

Provides six plots for each covariate index; histogram, path plot and acf plot for beta and for its subgradient.

Usage

## S3 method for class 'MHLS'
plot(x, index = 1:ncol(x$beta), skipS = FALSE, ...)

plot.MHLS  \hspace{1cm} \textit{Plot Metropolis-Hastings sampler outputs}
Arguments

- `x`: an object of class "MHLS", which is an output of `MHLS`.
- `index`: an index of covariates to plot.
- `skipS`: logical. If `skipS = TRUE`, plots beta only.
- `...`: additional arguments passed to or from other methods.

Details

`plot.MHLS` provides summary plots of beta and subgradient. The first column provides histogram of beta and subgradient, while the second and the third columns provide path and acf plots, respectively. If `skipS = TRUE`, this function provides summary plots for beta only.

Examples

```r
# set.seed(123)
set.seed(123)
n <- 10
p <- 5
X <- matrix(rnorm(n * p), n)
Y <- X %*% rep(1, p) + rnorm(n)
sigma2 <- 1
lbd <- .37
weights <- rep(1, p)
LassoResult <- lassoFit(X = X, Y = Y, lbd = lbd, type="lasso", weights = weights)
B0 <- LassoResult$B0
S0 <- LassoResult$S0
plot(MHLS(X = X, PE = rep(0, p), sig2 = 1, lbd = 1, group = 1:p, weights = weights, B0 = B0, S0 = S0, niter = 50, burnin = 0, type = "coeff"))
```

postInference.MHLS Post-inference with lasso estimator

Description

Provides confidence intervals for the set of active coefficients of lasso using Metropolis-Hastings sampler.

Usage

```r
postInference.MHLS(X, Y, lbd, weights = rep(1, ncol(X)), tau = rep(1, ncol(X)), sig2.hat, alpha = 0.05, nChain = 10, method, niterPerChain = 500, parallel = FALSE, ncores = 2L, returnSamples = FALSE, ...)
```
postInference.MHLS

Arguments

\textbf{X} \hspace{1em} \text{predictor matrix.}
\textbf{Y} \hspace{1em} \text{response vector.}
\textbf{lbd} \hspace{1em} \text{penalty term of lasso. By letting this argument be "cv.1se" or "cv.min", users can have the cross-validated lambda that gives either minimum squared error or that is within 1 std error bound.}
\textbf{weights} \hspace{1em} \text{weight vector with length equal to the number of coefficients. Default is \text{rep}(1, ncol(X)).}
\textbf{tau} \hspace{1em} \text{numeric vector. Standard deviation of proposal distribution for each beta. Adjust the value to get relevant level of acceptance rate. Default is \text{rep}(1, ncol(X)).}
\textbf{sig2.hat} \hspace{1em} \text{variance of error term.}
\textbf{alpha} \hspace{1em} \text{confidence level for confidence interval.}
\textbf{nChain} \hspace{1em} \text{the number of chains. For each chain, different plug-in beta will be generated from its confidence region.}
\textbf{method} \hspace{1em} \text{Type of robust method. Users can choose either "coeff" or "mu".}
\textbf{niterPerChain} \hspace{1em} \text{the number of iterations per chain.}
\textbf{parallel} \hspace{1em} \text{logical. If parallel = TRUE, uses parallelization. Default is parallel = FALSE.}
\textbf{ncores} \hspace{1em} \text{integer. The number of cores to use for parallelization.}
\textbf{returnSamples} \hspace{1em} \text{logical. If returnSamples = TRUE, print Metropolis-Hastings samples.}
\ldots \hspace{1em} \text{auxiliary \texttt{MHLS} arguments.}

Details

This function provides post-selection inference for the active coefficients selected by lasso. Uses Metropolis-Hastings sampler with multiple chains to draw from the distribution under a fixed active set and generates \((1-alpha)\) confidence interval for each active coefficients. Set \text{returnSamples} = \text{TRUE} to check the Metropolis-Hastings samples. Check the acceptance rate and adjust \text{tau} accordingly. We recommend to set \text{nChain} >= 10 and \text{niterPerChain} >= 500.

Value

\textbf{MHsamples} \hspace{1em} \text{a list of class \texttt{MHLS}.}
\textbf{confidenceInterval} \hspace{1em} \text{(1-alpha) confidence interval for each active coefficient.}

Examples

\begin{verbatim}
set.seed(123)
n <- 6
p <- 10
X <- matrix(rnorm(n*p),n)
Y <- X %*% rep(1,p) + rnorm(n)
sig2 <- 1
lbd <- .37
weights <- rep(1,p)
\end{verbatim}
parallel <- (.Platform$OS.type != "windows")
postInference.MHLS(X = X, Y = Y, lbd = lbd, sig2.hat = 1, alpha = .05,
nChain = 3, niterPerChain = 20, method = "coeff", parallel = parallel)
postInference.MHLS(X = X, Y = Y, lbd = lbd, sig2.hat = 1, alpha = .05,
nChain = 3, niterPerChain = 20, method = "coeff", parallel = parallel, returnSamples = TRUE)
postInference.MHLS(X = X, Y = Y, lbd = lbd, sig2.hat = 1, alpha = .05,
nChain = 3, niterPerChain = 20, method = "mu", parallel = parallel)
postInference.MHLS(X = X, Y = Y, lbd = lbd, sig2.hat = 1, alpha = .05,
nChain = 3, niterPerChain = 20, method = "mu", parallel = parallel, returnSamples = TRUE)

print.MHLS

**Print Metropolis-Hastings sampler outputs**

Description

Print a brief summary of the MH sampler outputs.

Usage

```r
## S3 method for class 'MHLS'
print(x, ...)
```

Arguments

- `x` an object of class "MHLS", which is a result of `MHLS`.
- `...` additional print arguments.

Details

`print.MHLS` prints out last 10 iterations and a brief summary of the simulation; number of iterations, number of burn-in periods, PE, PEtype and acceptance rate.

Value

Above results are silently returned.

Examples

```r
set.seed(123)
n <- 10
p <- 5
X <- matrix(rnorm(n * p), n)
Y <- X %*% rep(1, p) + rnorm(n)
sigma2 <- 1
lbd <- .37
weights <- rep(1, p)
LassoResult <- lassoFit(X = X, Y = Y, lbd = lbd, type="lasso", weights = weights)
B0 <- LassoResult$B0
S0 <- LassoResult$S0
```
Result <- MHLS(X = X, PE = rep(0, p), sig2 = sigma2, lbd = lbd, group = 1:p, 
weights = weights, B0 = B0, S0 = S0, niter = 50, burnin = 0, 
type = "coeff")
print(Result)

summary.MHLS

Summarizing Metropolis-Hastings sampler outputs

Description

Summary method for class "MHLS".

Usage

## S3 method for class 'MHLS'
summary(object, ...)

Arguments

object
an object of class "MHLS", which is a result of MHLS.

... additional arguments affecting the summary produced.

Details

This function provides a summary of each sampled beta and subgradient.

Value

mean, median, standard deviation, 2.5% quantile and 97.5% quantile for each beta and its subgradient.

Examples

#' set.seed(123)
n <- 10
p <- 5
X <- matrix(rnorm(n * p), n)
Y <- X %*% rep(1, p) + rnorm(n)
sigma2 <- 1
lbd <- .37
weights <- rep(1, p)
LassoResult <- lassoFit(X = X, Y = Y, lbd = lbd, type = "lasso", weights = weights)
B0 <- LassoResult$B0
S0 <- LassoResult$S0
summary(MHLS(X = X, PE = rep(0, p), sig2 = sigma2, lbd = lbd, 
weights = weights, B0 = B0, S0 = S0, niter = 50, burnin = 0, 
type = "coeff"))
Index

cv.lasso, 2, 6

hdIS, 3

lassoFit, 5

MHLS, 7, 8, 13–16

PB.CI, 9, 10
PBsampler, 3, 4, 9, 10
plot.MHLS, 8, 12, 13
postInference.MHLS, 13
print.MHLS, 15, 15

summary.MHLS, 8, 16