Package ‘BBSSL’

October 12, 2022

Title Bayesian Bootstrap Spike-and-Slab LASSO

Version 0.1.0

Description Posterior sampling for Spike-and-
Slab LASSO prior in linear models from Nie and Rockova <arXiv:2011.14279>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.1

Imports svMisc (>= 1.1.0), mvnfast (>= 0.2.5), rmutil (>= 1.1.3),
greybox (>= 0.5.1), statmod (>= 1.4.30), Matrix (>= 1.2-17),
glmnet (>= 2.0-16), truncnorm (>= 1.0-8), methods

NeedsCompilation yes

Author Lizhen Nie [aut, cre],
Veronika Rockova [aut]

Maintainer Lizhen Nie <lizhen@uchicago.edu>

Repository CRAN

Date/Publication 2021-04-27 08:10:05 UTC

R topics documented:

BB_SSL *BB-SSL*

Description

This function runs BB-SSL, WBB with fixed prior weight, and WBB with random prior weight. It solves the optimization by calling function SSLASSO_2, a variant of the function SSLASSO in CRAN package 'SSLASSO': in the version used, we do NOT standardize the design matrix and allow inputting initial values of beta's.

Usage

```
BB_SSL(y, X, method = 3, lambda, NSample, a, b, maxiter=500, eps = 1e-3, burn.in = FALSE,
length.out = 50, discard = FALSE, alpha = 3, sigma = 1, initial.beta,
penalty = "adaptive", theta=0.5)
```

Arguments

`y`	A vector of continuous responses (n x 1).
`X`	The design matrix (n x p), without an intercept.
`method`	A number between c(1,2,3) to specify which method to run, method = 1 is fixed WBB, method = 2 is random WBB, method = 3 is BB-SSL.
`lambda`	A two-dim vector = c(lambda0, lambda1).
`NSample`	An integer which specifies the number of samples to be generated.
`a, b`	Parameters of the prior.
`maxiter`	An integer which specifies the maximum number of iterations for SSLASSO_2 (default maxiter= 500).
`eps`	Convergence criterion when running SSLASSO_2: converged when difference in regression coefficients is less than eps (default eps = 0.001).
`burn.in`	A boolean to specify whether to use annealing on a sequence of lambda0's (default burn.in = FALSE).
`length.out`	An integer to specify the length of sequence of lambda0's used in annealing. This value is not used when burn.in = FALSE. Default is 50.
`discard`	A boolean to specify whether to discard unconverged sample points.
`alpha`	The parameter for generating weights in BB-SSL, which follows n x Dirichlet(alpha,...,alpha). Default is 3.
`sigma`	Noise standard deviation.
`initial.beta`	A vector of initial values of beta to used when solving SSLASSO_2 (n x 1).
`penalty`	The penalty (prior) to be applied to the model. Either "separable" (with a fixed theta) or "adaptive" (with a random theta, where theta ~ B(a,p)). The default is "adaptive".
`theta`	Prior mixing proportion. For "separable" penalty, this value is fixed. For "adaptive" penalty, this value is used as a starting value. Default is 0.5.

Value

A list of matrices, including matrix beta (NSample x p) and matrix gamma (NSample x p).

Author(s)

Lizhen Nie lizhen@statistics.uchicago.edu, Veronika Rockova Veronika.Rockova@chicagobooth.edu

References

Nie, L., & Ročková, V. (2020). Bayesian Bootstrap Spike-and-Slab LASSO. arXiv:2011.14279.

Newton, M. A., Polson, N. G., and Xu, J. (2020). Weighted Bayesian bootstrap for scalable posterior distributions. Canadian Journal of Statistics (In Press).

Examples

```
# -------------- Generate Data --------------
n = 50; p = 12;
truth.beta = c(1.3, 1.3, 1.3, 1.3);
truth.sigma = 1
data = Generate_data(truth.beta, p, n, truth.sigma = 1, rho = 0.6,"block",4)
y = data$y; X = data$X; beta = data$beta

# --------------- set parameters -----------------
lambda0 = 7; lambda1 = 0.15; lambda = c(lambda0, lambda1)
a = 1; b = p #beta prior for theta



#--------------- BB-SSL -------------
# this is for demonstration of usage only
# in practice, you may want to use more iterations!
BB.SSL.result = BB_SSL(y, X, method = 3, lambda = c(lambda0, lambda1), NSample = 100, a, b,
maxiter = 500, length.out = 50, burn.in = FALSE, discard = TRUE, alpha=1,
initial.beta = rep(0,p))

# Alternatively, you can use SSLASSO_2 solution to get an initial value of beta's
result = SSLASSO_2(X, y, penalty = "adaptive", variance = "fixed", sigma = truth.sigma,
                   lambda1 = lambda1, lambda0 = seq(lambda1, lambda0, length.out = 50),
                   a = a, b = b,
                   max.iter = 500, initial.beta = rep(0,p))

fixed.WBB.result = BB_SSL(y, X, method = 1, lambda = c(lambda0, lambda1), NSample = 100,
                          a, b, maxiter = 500, length.out = 50, burn.in = FALSE,
                          discard = TRUE, initial.beta = result$beta[,50])

random.WBB.result = BB_SSL(y, X, method = 2, lambda = c(lambda0, lambda1), NSample = 100,
                           a, b, maxiter = 500, length.out = 50, burn.in = FALSE,
                           discard = TRUE, initial.beta = result$beta[,50])

BB.SSL.result = BB_SSL(y, X, method = 3, lambda = c(lambda0, lambda1), NSample = 100, a,
                      b, maxiter = 500, length.out = 50, burn.in = FALSE, discard = TRUE,
alpha=1, initial.beta = result$beta[,50])
```

Generate_data *Generate_data*

Description

This function generates data (X,y) with specified correlation and noise standard deviation.

Usage

```
Generate_data(truth.beta, p, n, truth.sigma, rho, correlation = c("block", "all"),
NumOfBlock)
```

Arguments

`truth.beta`	A vector of active beta's (s x 1, with s being the number of active coordinates).
`p`	The number of covariates.
`n`	The number of observations.
`truth.sigma`	Noise standard deviation.
`rho`	Correlation Coefficient.
`correlation`	Correlation structure. Correlation = "block" means predictors are grouped into equi-size blocks where each block contains one active predictor, and the within-block correlation coefficient is rho; predictors in different blocks are mutually independent. Correlation = "all" means all predictors are equi-correlated with coefficient rho.
`NumOfBlock`	Number of blocks, used only when correlation = 'block'.

Value

A list, including vector 'y' (n x 1), matrix 'X' (n x p), vector 'beta' (p x 1).

Gibbs *Gibbs*

Description

This function runs SSVS for linear regression with Spike-and-Slab LASSO prior. By default, this function uses the speed-up trick in Bhattacharya et al. (2016) when p > n.

Usage

```
Gibbs(y, X, a, b, lambda, maxiter, burn.in, initial.beta = NULL, sigma = 1)
```

Arguments

`y`	A vector of continuous responses (n x 1).
`X`	The design matrix (n x p), without an intercept.
`a, b`	Parameters of the prior.
`lambda`	A two-dim vector = c(lambda0, lambda1).
`maxiter`	An integer which specifies the maximum number of iterations for MCMC.
`burn.in`	An integer which specifies the maximum number of burn-in iterations for MCMC.
`initial.beta`	A vector of initial values of beta to used. If set to NULL, the LASSO solution with 10-fold cross validation is used. Default is NULL.
`sigma`	Noise standard deviation. Default is 1.

Value

A list, including matrix 'beta' ((maxiter-burn.in) x p), matrix 'tau2' ((maxiter-burn.in) x p), matrix 'gamma' ((maxiter-burn.in) x p), vector 'theta' ((maxiter-burn.in) x 1).

Author(s)

Lizhen Nie lizhen@statistics.uchicago.edu, Veronika Rockova Veronika.Rockova@chicagobooth.edu

References

Nie, L., & Ročková, V. (2020). Bayesian Bootstrap Spike-and-Slab LASSO. arXiv:2011.14279.

Bhattacharya, A., Chakraborty, A., & Mallick, B. K. (2016). Fast sampling with Gaussian scale mixture priors in high-dimensional regression. Biometrika, 103(4):985.

Examples

```
n = 50; p = 12;
truth.beta = c(1.3, 1.3, 1.3, 1.3);
truth.sigma = 1
data = Generate_data(truth.beta, p, n, truth.sigma = 1, rho = 0.6,"block",4)
y = data$y; X = data$X; beta = data$beta

# --------------- set parameters -----------------
lambda0 = 7; lambda1 = 0.15; lambda = c(lambda0, lambda1)
a = 1; b = p #beta prior for theta

# this is for demonstration of usage only
# in practice, you may want to use more iterations!
MCchain1 = Gibbs(y, X, a, b, lambda, maxiter = 1000, burn.in = 100)
```

Gibbs2 *Gibbs2*

Description

This function runs one-site Gibbs sampler for linear regression with Spike-and-Slab LASSO prior.

Usage

```
Gibbs2(y, X, a, b, lambda, maxiter, burn.in, initial.beta = NULL, sigma = 1)
```

Arguments

`y`	A vector of continuous responses (n x 1).
`X`	The design matrix (n x p), without an intercept.
`a, b`	Parameters of the prior.
`lambda`	A two-dim vector = c(lambda0, lambda1).
`maxiter`	An integer which specifies the maximum number of iterations for MCMC.
`burn.in`	An integer which specifies the number of burn-in iterations for MCMC.
`initial.beta`	A vector of initial values of beta to used. If set to NULL, the LASSO solution with 10-fold cross validation is used. Default is NULL.
`sigma`	Noise standard deviation. Default is 1.

Value

A list, including matrix beta ((maxiter-burn.in) x p) and matrix gamma (maxiter-burn.in) x p, vector theta ((maxiter-burn.in) x 1)

Author(s)

Lizhen Nie lizhen@statistics.uchicago.edu, Veronika Rockova Veronika.Rockova@chicagobooth.edu

References

Nie, L., & Ročková, V. (2020). Bayesian Bootstrap Spike-and-Slab LASSO. arXiv:2011.14279.

Examples

```
n = 50; p = 12;
truth.beta = c(1.3, 1.3, 1.3, 1.3);
truth.sigma = 1
data = Generate_data(truth.beta, p, n, truth.sigma = 1, rho = 0.6,"block",4)
y = data$y; X = data$X; beta = data$beta

# --------------- set parameters -----------------
lambda0 = 7; lambda1 = 0.15; lambda = c(lambda0, lambda1)
```

```
a = 1; b = p #beta prior for theta

# this is for demonstration of usage only
# in practice, you may want to use more iterations!
MCchain2 = Gibbs2(y, X, a, b, lambda, maxiter = 1000, burn.in = 100)
```

plot.SSLASSO *Plot coefficients from a "SSLASSO" object*

Description

Produces a plot of the coefficient paths for a fitted "SSLASSO" object.

Usage

```
## S3 method for class 'SSLASSO'
plot(x, ...)
```

Arguments

x	Fitted "SSLASSO" model.
...	Other graphical parameters to plot.

Value

A plot of the coefficient paths for a fitted "SSLASSO" object.

References

Rockova, V. and George, E.I. (2018) The Spike-and-Slab LASSO. Journal of the American Statistical Association.

See Also

SSLASSO_2

Examples

```
## Linear regression, where p>n

n = 100; p = 1000;
truth.beta = c(2, 3, -3, 4); # high-dimensional case
truth.sigma = 1
data = Generate_data(truth.beta, p, n, truth.sigma = 1, rho = 0.6,"all", 4)
y = data$y; X = data$X; beta = data$beta

# --------------- set parameters -----------------
lambda0 = 50; lambda1 = 0.05; lambda = c(lambda0, lambda1)
```

```
a = 1; b = p #beta prior for theta


# Separable penalty with fixed theta

result<-SSLASSO_2(X, y,penalty="separable", variance = "fixed",
lambda1 = lambda1, lambda0 = seq(from=lambda1,to=lambda0,length.out=50),
theta = 4/p,initial.beta = rep(0,p))

plot(result)
```

SSLASSO_2 *The Spike-and-Slab LASSO (for BB-SSL).*

Description

Spike-and-Slab LASSO is a spike-and-slab refinement of the LASSO procedure, using a mixture of Laplace priors indexed by `lambda0` (spike) and `lambda1` (slab).

The `SSLASSO` procedure fits coefficients paths for Spike-and-Slab LASSO-penalized linear regression models over a grid of values for the regularization parameter `lambda0`. The code has been adapted from the `SSLASSO` package (Rockova, V. and Moran, G. (2019). Package ‘SSLASSO’.) such that now it does NOT normalize each column and allows specifying initialization value).

Usage

```
SSLASSO_2(X, y, initial.beta, penalty = c("adaptive", "separable"),
variance = c("fixed", "unknown"), lambda1, lambda0, nlambda = 100,
theta = 0.5, sigma = 1, a = 1, b, eps = 0.001, max.iter = 500,
counter = 10, warn = FALSE)
```

Arguments

`X`	The design matrix (n x p), without an intercept. `SSLASSO` standardizes the data by default.
`y`	Vector of continuous responses (n x 1). The responses will be centered by default.
`initial.beta`	Initial value for beta when searching for the solution.
`penalty`	The penalty to be applied to the model. Either "separable" (with a fixed `theta`) or "adaptive" (with a random `theta`, where `theta ~ B(a,p)`). The default is `"adaptive"`.
`variance`	Whether the error variance is also estimated. Either "fixed" (with a fixed `sigma`) or "unknown" (with a random `sigma`, where `p(sigma) ~ 1/sigma`). The default is `"fixed"`.
`lambda1`	Slab variance parameter. Needs to be less than `lambda0`. The default is `lambda0 = 1`.

`lambda0`	Spike penalty parameters (L x 1). Either a numeric value for a single run (L=1) or a sequence of increasing values for dynamic posterior exploration. The default is `lambda0 = seq(1, nrow(X), length.out = 100)`.
`nlambda`	The number of `lambda0` values. Default is 100.
`theta`	Prior mixing proportion. For "separable" penalty, this value is fixed. For "adaptive" penalty, this value is used as a starting value.
`sigma`	Error variance. For "fixed" variance, this value is fixed. For "unknown" variance, this value is used as a starting value.
`a`	Hyperparameter of the beta prior `B(a,b)` for the adaptive penalty (default `a = 1`).
`b`	Hyperparameter of the beta prior `B(a,b)` for the adaptive penalty (default `b = ncol(X)`).
`eps`	Convergence criterion: converged when difference in regression coefficients is less than `eps` (default `eps = 0.001`).
`max.iter`	Maximum number of iterations. Default is 500.
`counter`	Applicable only for the adaptive penalty. Determines how often the parameter `theta` is updated throughout the cycles of coordinate ascent. Default is 10.
`warn`	TRUE if warnings should be printed; FALSE by default

Details

The sequence of models indexed by the regularization parameter `lambda0` is fitted using a coordinate descent algorithm. The algorithm uses screening rules for discarding irrelevant predictors along the lines of Breheny (2011).

Value

An object with S3 class `"SSLASSO"` containing:

`beta`	The fitted matrix of coefficients (p x L). The number of rows is equal to the number of coefficients `p`, and the number of columns is equal to `L` (the length of `lambda0`).
`intercept`	A vector of length `L` containing the intercept for each value of `lambda0`. The intercept is `intercept = mean(y) - crossprod(XX, beta)`, where `XX` is the centered design matrix.
`iter`	A vector of length `L` containing the number of iterations until convergence at each value of `lambda0`.
`lambda0`	The sequence of regularization parameter values in the path.
`penalty`	Same as above.
`thetas`	A vector of length `L` containing the hyper-parameter values `theta` (the same as `theta` for "separable" penalty).
`sigmas`	A vector of length `L` containing the values `sigma` (the same as the initial `sigma` for "known" variance).
`select`	A (p x L) binary matrix indicating which variables were selected along the solution path.
`model`	A single model chosen after the stabilization of the regularization path.

References

Rockova, V. and George, E.I. (2018) The Spike-and-Slab LASSO. Journal of the American Statistical Association.

Moran, G., Rockova, V. and George, E.I. (2018) On variance estimation for Bayesian variable selection. <https://arxiv.org/abs/1801.03019>.

Nie, L., & Ročková, V. (2020). Bayesian Bootstrap Spike-and-Slab LASSO. arXiv:2011.14279.

Examples

```
## Linear regression, where p > n

p <- 1000
n <- 100

X <- matrix(rnorm(n*p), nrow = n, ncol = p)
beta <- c(1, 2, 3, rep(0, p-3))
y = X[,1] * beta[1] + X[,2] * beta[2] + X[,3] * beta[3] + rnorm(n)

# Oracle SSLASSO with known variance

result1 <- SSLASSO_2(X, y, penalty = "separable", theta = 3/p, initial.beta = rep(0,p))
plot(result1)

# Adaptive SSLASSO with known variance

result2 <- SSLASSO_2(X, y, initial.beta = rep(0,p))
plot(result2)
```

Index