Package 'Mhorseshoe'

November 24, 2023

Title Approximate Algorithm for Horseshoe Prior

Version 0.1.2

Description Provides an approximate algorithm for the horseshoe estimator used in Bayesian linear models. By implementing a sampler with high computational cost in the 'Rcpp' package and using an approximate algorithm that reduces matrix calculation complexity, parameter estimation speed for high-dimensional sparse data is faster. The approximate algorithm is described in Johndrow et al. (2020) <https://www.jmlr.org/papers/v21/19-536.html>.

Encoding UTF-8

Imports stats, Rcpp (>= 1.0.11)

LinkingTo Rcpp

RoxygenNote 7.2.3

License MIT + file LICENSE

Suggests knitr, rmarkdown, ggplot2, horseshoe

VignetteBuilder knitr

NeedsCompilation yes

Author Kang Mingi [aut, cre],
Lee Kyoungjae [aut]

Maintainer Kang Mingi <leehuimin115@g.skku.edu>

Repository CRAN

Date/Publication 2023-11-24 09:20:10 UTC

R topics documented:

`approx_horseshoe` *Run approximate MCMC algorithm for horseshoe prior*

Description

In this function, The algorithm introduced in Section 2.2 of Johndrow et al. (2020) is implemented, and is a horseshoe estimator that generally considers the case where $p >> N$. The assumptions and notations for the model are the same as those in `Mhorseshoe`. This algorithm introduces a threshold and uses only a portion of the total p columns for matrix multiplication, lowering the computational cost compared to the existing horseshoe estimator. According to Section 3.2 of Johndrow et al. (2020), the approximate MCMC algorithm applying the methodology constructs an approximate Markov chain P_ϵ that can converge to an exact Markov chain P, and acceptable results were confirmed through empirical analysis of simulation and real data. The "auto.threshold" argument in this function is an adaptive probability algorithm for threshold developed in this package, which is an algorithm that estimates and updates a new threshold through updated shrinkage parameters.

Usage

```
approx_horseshoe(
  X,
  y,
  burn = 1000,
  iter = 5000,
  auto.threshold = TRUE,
  threshold = 0,
  tau = 1,
  s = 0.8,
  sigma2 = 1,
  w = 1,
  alpha = 0.05,
  a = 0.2,
  b = 10,
  t = 10,
  adapt_p0 = 0,
  adapt_p1 = -4.6 * 10^(-4)
)
```

Arguments

`X`	Design matrix, $X \in \mathbb{R}^{N \times p}$.
`y`	Response vector, $y \in \mathbb{R}^N$.
`burn`	Number of burn-in samples. Default is 1000.
`iter`	Number of samples to be drawn from the posterior. Default is 5000.
`auto.threshold`	Argument for setting whether to use an algorithm that automatically updates the threshold using adaptive probability.

threshold	Threshold to be used in the approximate MCMC algorithm. If you select auto.threshold = FALSE, and threshold = 0(This is the default value for the threshold argument), the threshold is set according to the sizes of N and p. if $p < N$, $\delta = 1/\sqrt{Np}$, else $\delta = 1/p$. Or, you can set your custom value directly through this argument. For more information about δ, see Mhorseshoe and 4.1 of Johndrow et al. (2020).
tau	Initial value of the global shrinkage parameter τ when starting the algorithm. Default is 1.
s	s^2 is the variance of tau's MH proposal distribution. 0.8 is a good default. If set to 0, the algorithm proceeds by fixing the global shrinkage parameter τ to the initial setting value.
sigma2	error variance σ^2. Default is 1.
w	Parameter of gamma prior for σ^2. Default is 1.
alpha	$100(1-\alpha)\%$ credible interval setting argument.
a	Parameter of the rejection sampler, and it is recommended to leave it at the default value, $a = 1/5$.
b	Parameter of the rejection sampler, and it is recommended to leave it at the default value, $b = 10$.
t	Threshold update cycle for adaptive probability algorithm when auto.threshold is set to TRUE. default is 10.
adapt_p0	Parameter p_0 of adaptive probability, $p(t) = exp[p_0 + p_1 t]$. default is 0.
adapt_p1	Parameter a_1 of adaptive probability, $p(t) = exp[p_0 + p_1 t]$. default is -4.6×10^{-4}.

Value

BetaHat	Posterior mean of β.
LeftCI	Lower bound of $100(1-\alpha)\%$ credible interval for β.
RightCI	Upper bound of $100(1-\alpha)\%$ credible interval for β.
Sigma2Hat	Posterior mean of σ^2.
TauHat	Posterior mean of τ.
LambdaHat	Posterior mean of $\lambda_j,\ j = 1, 2, ...p$..
ActiveMean	Average number of elements in the active set per iteration in this algorithm.
BetaSamples	Samples from the posterior of β.
LambdaSamples	Lambda samples through rejection sampling.
TauSamples	Tau samples through MH algorithm.
Sigma2Samples	Samples from the posterior of the parameter $sigma^2$.
ActiveSet	Matrix indicating active elements as 1 and non-active elements as 0 per iteration of the MCMC algorithm.

Approximate algorithm details

Approximate algorithm has the following changes:

$$D_\delta = diag\left(\eta_j^{-1} 1\left(\xi^{-1}\eta_j^{-1} > \delta,\ j = 1, 2, ..., p.\right)\right),$$
$$M_\xi \approx M_{\xi,\delta} = I_N + \xi^{-1} X D_\delta X^T,$$

Where $\eta_j = \lambda_j^{-2}$, λ_j are local shrinkage parameters, $\xi = \tau^{-2}$, τ is a global shrinkage parameter, $1(\cdot)$ is an indicator function that returns 1 if the conditions in the parentheses are satisfied, and 0 otherwise, and δ is the threshold. The set of X's columns: $\{x_j\ :\ \xi^{-1}\eta_j^{-1} > \delta,\ j = 1, 2, ..., p\}$ is defined as the active set, and let's define S as the index set of the active set:

$$S = \{j \mid \xi^{-1}\eta_j^{-1} > \delta,\ j = 1, 2, ..., p.\}.$$

Recalling the posterior distribution for β, it is as follows:

$$\beta | y, X, \eta, \xi, \sigma \sim N\left(\left(X^T X + \left(\xi^{-1} D\right)^{-1}\right)^{-1} X^T y,\ \sigma^2 \left(X^T X + \left(\xi^{-1} D\right)^{-1}\right)^{-1}\right).$$

If $\xi^{-1}\eta_j^{-1}$ is very small, the posterior of β will have a mean and variance close to 0. Therefore, let's set $\xi^{-1}\eta_j^{-1}$ smaller than δ to 0 and the size of inverse $M_{\xi,\delta}$ matrix is reduced as follows.

$$length(S) = s_\delta \leq p, X_S \in R^{N \times s_\delta}, \quad D_S \in R^{s_\delta \times s_\delta}, M_{\xi,\delta}^{-1} = \left(I_N + \xi^{-1} X_S D_S X_S^T\right)^{-1}.$$

$M_{\xi,\delta}^{-1}$ can be expressed using the Woodbury identity as follows.

$$M_{\xi,\delta}^{-1} = I_N - X_S \left(\xi D_S^{-1} + X_S^T X_S\right)^{-1} X_S^T.$$

$M_{\xi,\delta}^{-1}$, which reduces the computational cost, is applied to all parts of this algorithm, β samples are extracted from the posterior using fast sampling(Bhattacharya et al.,2016) as follows.

$$u \sim N_p(0, \xi^{-1} D), \quad f \sim N_N(0, I_N), v = Xu + f, \quad v^\star = M_{\xi,\delta}^{-1}(y/\sigma - v), \beta = \sigma(u + \xi^{-1} D_\delta X^T v^\star).$$

Adaptive probability algorithm for threshold update

If the auto.threshold argument is set to TRUE, this algorithm operates every t iteration to estimate the threshold and decide whether to update. In this algorithm, the process of updating a new threshold is added by applying the properties of the shrinkage weight $k_j,\ j = 1, 2, ..., p$ proposed by Piironen and Vehtari (2017). In the prior of $\beta_j \sim N(0, \sigma^2\tau^2\lambda_j^2) = N(0, \sigma^2\xi^{-1}\eta_j^{-1})$, the variable m_{eff} is defined as follows.

$$k_j = 1/\left(1 + n\xi^{-1} s_j^2 \eta_j^{-1}\right), \quad j = 1, 2, ..., p, m_{eff} = \sum_{j=1}^{p} (1 - k_j).$$

The assumptions and notations for the model are the same as those in Mhorseshoe, and $s_j,\ j = 1, 2, ..., p$ are the diagonal components of $X^T X$. For the zero components of β, k_j is derived close to 1, and nonzero's k_j is derived close to 0, so the variable m_{eff}, the sum of $1 - k_j$, is called the effective number of nonzero coefficients. In this algorithm, the threshold δ is updated to set $s_\delta = ceiling(m_{eff})$.

Adaptive probability is defined to satisfy Theorem 5(diminishing adaptation condition) of Roberts and Rosenthal (2007). at Tth iteration,

$$p(T) = exp[p_0 + p_1 T], \quad p_1 < 0, \quad u \sim U(0,1), if\ u < p(T),\ update\ \delta\ so\ that\ s_\delta = ceiling(m_{eff}).$$

The default is $p_0 = 0$, $p_1 = -4.6 \times 10^{-4}$, and under this condition, $p(10000) < 0.01$ is satisfied.

References

Bhattacharya, A., Chakraborty, A., & Mallick, B. K. (2016). Fast sampling with Gaussian scale mixture priors in high-dimensional regression. Biometrika, asw042.

Johndrow, J., Orenstein, P., & Bhattacharya, A. (2020). Scalable Approximate MCMC Algorithms for the Horseshoe Prior. In Journal of Machine Learning Research (Vol. 21).

Piironen, J., & Vehtari, A. (2017). Sparsity information and regularization in the horseshoe and other shrinkage priors. Electronic Journal of Statistics, 11, 5018-5051.

Roberts G, Rosenthal J. Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms. J Appl Prob. 2007;44:458–475.

Examples

```
# Making simulation data.
set.seed(123)
N <- 200
p <- 100
true_beta <- c(rep(1, 10), rep(0, 90))

X <- matrix(1, nrow = N, ncol = p) # Design matrix X.
for (i in 1:p) {
  X[, i] <- stats::rnorm(N, mean = 0, sd = 1)
}

y <- vector(mode = "numeric", length = N) # Response variable y.
e <- rnorm(N, mean = 0, sd = 2) # error term e.
for (i in 1:10) {
  y <- y + true_beta[i] * X[, i]
}
y <- y + e

# Run with auto.threshold option
result1 <- approx_horseshoe(X, y, burn = 0, iter = 100)

# Run with fixed custom threshold
result2 <- approx_horseshoe(X, y, burn = 0, iter = 100,
```

```
                              auto.threshold = FALSE, threshold = 1/(5 * p))

# posterior mean
betahat <- result1$BetaHat

# Lower bound of the 95% credible interval
leftCI <- result1$LeftCI

# Upper bound of the 95% credible interval
RightCI <- result1$RightCI
```

`exact_horseshoe` *Run exact MCMC algorithm for horseshoe prior*

Description

The exact MCMC algorithm introduced in Section 2.1 of Johndrow et al. (2020) was implemented in this function. This algorithm is the horseshoe estimator that updates the global shrinkage parameter τ using Metropolis-Hastings algorithm, and uses blocked-Gibbs sampling for (τ, β, σ). The local shrinkage parameter $\lambda_j,\ j = 1, 2, ..., p$ is updated by the rejection sampler.

Usage

```
exact_horseshoe(
  X,
  y,
  burn = 1000,
  iter = 5000,
  a = 1/5,
  b = 10,
  s = 0.8,
  tau = 1,
  sigma2 = 1,
  w = 1,
  alpha = 0.05
)
```

Arguments

`X`	Design matrix, $X \in \mathbb{R}^{N \times p}$.
`y`	Response vector, $y \in \mathbb{R}^N$.
`burn`	Number of burn-in samples. Default is 1000.
`iter`	Number of samples to be drawn from the posterior. Default is 5000.
`a`	Parameter of the rejection sampler, and it is recommended to leave it at the default value, $a = 1/5$.

b	Parameter of the rejection sampler, and it is recommended to leave it at the default value, $b = 10$.
s	s^2 is the variance of tau's MH proposal distribution. 0.8 is a good default. If set to 0, the algorithm proceeds by fixing the global shrinkage parameter τ to the initial setting value.
tau	Initial value of the global shrinkage parameter τ when starting the algorithm. Default is 1.
sigma2	error variance σ^2. Default is 1.
w	Parameter of gamma prior for σ^2. Default is 1.
alpha	$100(1-\alpha)\%$ credible interval setting argument.

Details

See Mhorseshoe or browseVignettes("Mhorseshoe").

Value

BetaHat	Posterior mean of β.
LeftCI	Lower bound of $100(1-\alpha)\%$ credible interval for β.
RightCI	Upper bound of $100(1-\alpha)\%$ credible interval for β.
Sigma2Hat	Posterior mean of σ^2.
TauHat	Posterior mean of τ.
LambdaHat	Posterior mean of $\lambda_j,\ j = 1, 2, ...p..$
BetaSamples	Samples from the posterior of β.
LambdaSamples	Lambda samples through rejection sampling.
TauSamples	Tau samples through MH algorithm.
Sigma2Samples	Samples from the posterior of the parameter $sigma^2$.

References

Johndrow, J., Orenstein, P., & Bhattacharya, A. (2020). Scalable Approximate MCMC Algorithms for the Horseshoe Prior. In Journal of Machine Learning Research (Vol. 21).

Examples

```
# Making simulation data.
set.seed(123)
N <- 50
p <- 100
true_beta <- c(rep(1, 10), rep(0, 90))

X <- matrix(1, nrow = N, ncol = p) # Design matrix X.
for (i in 1:p) {
  X[, i] <- stats::rnorm(N, mean = 0, sd = 1)
}
```

```
y <- vector(mode = "numeric", length = N) # Response variable y.
e <- rnorm(N, mean = 0, sd = 2) # error term e.
for (i in 1:10) {
  y <- y + true_beta[i] * X[, i]
}
y <- y + e

# Run
result <- exact_horseshoe(X, y, burn = 0, iter = 100)

# posterior mean
betahat <- result$BetaHat

# Lower bound of the 95% credible interval
leftCI <- result$LeftCI

# Upper bound of the 95% credible interval
RightCI <- result$RightCI
```

Index