Package ‘ramcmc’

October 6, 2021

Title Robust Adaptive Metropolis Algorithm

Version 0.1.2

Date 2021-10-06

Description Function for adapting the shape of the random walk Metropolis proposal
as specified by robust adaptive Metropolis algorithm by Vihola (2012) <doi:10.1007/s11222-011-9269-5>.
The package also includes fast functions for rank-one Cholesky update and downdate.
These functions can be used directly from R or the corresponding C++ header files
can be easily linked to other R packages.

License GPL (>= 2)

BugReports https://github.com/helske/ramcmc/issues

Suggests testthat, knitr, rmarkdown

Imports Rcpp (>= 0.12.8)

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 5.0.1

VignetteBuilder knitr

NeedsCompilation yes

Author Jouni Helske [aut, cre] (<https://orcid.org/0000-0001-7130-793X>)

Maintainer Jouni Helske <jouni.helske@iki.fi>

Repository CRAN

Date/Publication 2021-10-06 21:40:02 UTC

R topics documented:

`adapt_S` *Update the Proposal of RAM Algorithm*

Description

Given the lower triangular matrix S obtained from the Cholesky decomposition of the shape of the proposal distribution, function `adapt_S` updates S according to the RAM algorithm.

Usage

```
adapt_S(S, u, current, n, target = 0.234, gamma = 2/3)
```

Arguments

`S`	A lower triangular matrix corresponding to the Cholesky decomposition of the scale of the proposal distribution.
`u`	A vector with with length matching with the dimensions of S.
`current`	The current acceptance probability.
`n`	Scaling parameter corresponding to the current iteration number.
`target`	The target acceptance rate. Default is 0.234.
`gamma`	Scaling parameter. Default is 2/3.

Value

If the resulting matrix is positive definite, an updated value of S. Otherwise original S is returned.

Note

If the downdating would result non-positive definite matrix, no adaptation is performed.

References

Matti Vihola (2012). "Robust adaptive Metropolis algorithm with coerced acceptance rate". Statistics and Computing, 22: 997. doi:10.1007/s11222-011-9269-5

Examples

```
# sample from standard normal distribution
# use proposals from the uniform distribution on
# interval (-s, s), where we adapt s

adapt_mcmc <- function(n = 10000, s) {
  x <- numeric(n)
  loglik_old <- dnorm(x[1], log = TRUE)
  for (i in 2:n) {
    u <- s * runif(1, -1, 1)
```

```
    prop <- x[i] + u
    loglik <- dnorm(prop, log = TRUE)
    accept_prob <- min(1, exp(loglik - loglik_old))
    if (runif(1) < accept_prob) {
      x[i] <- prop
      loglik_old <- loglik
    } else {
      x[i] <- x[i - 1]
    }
    # Adapt only during the burn-in
    if (i < n/2) {
      s <- adapt_S(s, u, accept_prob, i)
    }
  }
  list(x = x[(n/2):n], s = s)
}

out <- adapt_mcmc(1e5, 2)
out$s
hist(out$x)
# acceptance rate:
1 / mean(rle(out$x)$lengths)
```

`chol_downdate` *Rank-one Downdate of Cholesky Decomposition*

Description

Given the lower triangular matrix L obtained from the Cholesky decomposition of A, function `chol_downdate` updates L such that it corresponds to the decomposition of A - u*u' (if such decomposition exists).

Usage

```
chol_downdate(L, u)
```

Arguments

`L`	A lower triangular matrix. Strictly upper diagonal part is not referenced.
`u`	A vector with with length matching with the dimensions of L.

Value

Updated L.

Note

The function does not check that the resulting matrix is positive semidefinite.

chol_update *Rank-one Update of Cholesky Decomposition*

Description

Given the lower triangular matrix L obtained from the Cholesky decomposition of A, function chol_update updates L such that it corresponds to the decomposition of A + u*u'.

Usage

```
chol_update(L, u)
```

Arguments

L	A lower triangular matrix. Strictly upper diagonal part is not referenced.
u	A vector with with length matching with the dimensions of L.

Value

Updated L.

Examples

```
L <- matrix(c(4,3,0,5), 2, 2)
u <- c(1, 2)
chol_update(L, u)
t(chol(L %*% t(L) + u %*% t(u)))
```

Index