

Package ‘hdtg’

August 7, 2022

Title Generate Samples from Multivariate Truncated Normal Distributions

Version 0.2.0

Maintainer Zhenyu Zhang <zhangzhenyusa@gmail.com>

Description Efficient sampling from high-dimensional truncated Gaussian distributions, or multivariate truncated normal (MTN). Techniques include zigzag Hamiltonian Monte Carlo as in Akihiko Nishimura, Zhenyu Zhang and Marc A. Suchard (2021) <[arXiv:2104.07694](https://arxiv.org/abs/2104.07694)>, and harmonic Monte in Ari Pakman and Liam Paninski (2014) <[doi:10.1080/10618600.2013.788448](https://doi.org/10.1080/10618600.2013.788448)>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.0

Imports Rcpp, RcppParallel, RcppXsimd, mgcv, stats, Rdpack

RdMacros Rdpack

LinkingTo Rcpp, RcppEigen, RcppParallel, RcppXsimd

SystemRequirements C++14

NeedsCompilation yes

Author Zhenyu Zhang [aut, cre],
Andrew Chin [aut],
Akihiko Nishimura [aut],
Marc A. Suchard [aut],
John W. Ratcliff et al. [cph, ctb] (authors and copyright holders of
see2neon.h under an MIT license)

Repository CRAN

Date/Publication 2022-08-07 00:30:02 UTC

R topics documented:

cholesky	2
createEngine	2
createNutsEngine	3

getInitialPosition	4
getZigzagSample	4
harmonicHMC	6
setMean	7
setPrecision	7
zigzagHMC	8

Index	10
--------------	-----------

cholesky	<i>Efficient Cholesky decomposition</i>
----------	---

Description

Compute Cholesky decomposition of a matrix.

Usage

```
cholesky(A)
```

Arguments

A matrix to decompose

Value

upper triangular matrix R such that $A = U^*U$.

createEngine	<i>Create a Zigzag-HMC engine object</i>
--------------	--

Description

Create the C++ object to set up SIMD vectorization for speeding up calculations for Zigzag-HMC ("Zigzag-HMC engine").

Usage

```
createEngine(
  dimension,
  lowerBounds,
  upperBounds,
  seed,
  mean,
  precision,
  flags = 128L
)
```

Arguments

dimension	the dimension of MTN.
lowerBounds	a vector specifying the lower bounds.
upperBounds	a vector specifying the upper bounds.
seed	random seed.
mean	the mean vector.
precision	the precision matrix.
flags	which SIMD instruction set to use. 128 = SSE, 256 = AVX.

Value

a list whose only element is the Zigzag-HMC engine object.

createNutsEngine	<i>Create a Zigzag-NUTS engine object</i>
------------------	---

Description

Create the C++ object to set up SIMD vectorization for speeding up calculations for Zigzag-NUTS ("Zigzag-NUTS engine").

Usage

```
createNutsEngine(
  dimension,
  lowerBounds,
  upperBounds,
  seed,
  stepSize,
  mean,
  precision,
  flags = 128L
)
```

Arguments

dimension	the dimension of MTN.
lowerBounds	a vector specifying the lower bounds.
upperBounds	a vector specifying the upper bounds.
seed	random seed.
stepSize	the base step size for Zigzag-NUTS.
mean	the mean vector.
precision	the precision matrix.
flags	which SIMD instruction set to use. 128 = SSE, 256 = AVX.

Value

a list whose only element is the Zigzag-NUTS engine object.

getInitialPosition	<i>Get an eligible initial value for a MTN with given mean and truncations</i>
--------------------	--

Description

For a given MTN the function returns an initial vector whose elements are one of: (1) middle point of the truncation interval if both lower and upper bounds are finite (2) lower (upper) bound +0.1 (-0.1) if only the lower (upper) bound is finite (3) the corresponding mean value if lower bound = -Inf are upper bound = Inf.

Usage

```
getInitialPosition(mean, lowerBounds, upperBounds)
```

Arguments

mean	a d-dimensional mean vector.
lowerBounds	a d-dimensional vector specifying the lower bounds.
upperBounds	a d-dimensional vector specifying the lower bounds.

Value

an eligible d-dimensional initial vector.

getZigzagSample	<i>Draw one MTN sample with Zigzag-HMC or Zigzag-NUTS</i>
-----------------	---

Description

Simulate the Zigzag-HMC or Zigzag-NUTS dynamics on a given MTN.

Usage

```
getZigzagSample(position, momentum = NULL, nutsFlg, engine, stepZZHMC = NULL)
```

Arguments

position	a d-dimensional initial position vector.
momentum	a d-dimensional initial momentum vector.
nutsFlg	logical. If TRUE the No-U-Turn sampler will be used (Zigzag-NUTS).
engine	list. Its engine element is a pointer to the Zigzag-HMC engine (or Zigzag-NUTS engine) C++ object that implements fast computations for Zigzag-HMC (or Zigzag-NUTS).
stepZZHMC	step size for Zigzag-HMC. If nutsFlg = TRUE, engine contains the base step size for Zigzag-NUTS).

Value

one MCMC sample from the target MTN.

Note

getZigzagSample is particularly efficient when the target MTN has a random mean and covariance/precision where one can reuse the Zigzag-HMC engine object while updating the mean and covariance. The following example demonstrates such a use.

Examples

```

set.seed(1)
n <- 1000
d <- 10
samples <- array(0, c(n, d))

# initialize MTN mean and precision
m <- rnorm(d, 0, 1)
prec <- rWishart(n = 1, df = d, Sigma = diag(d))[, , 1]
# call createEngine once
engine <- createEngine(dimension = d, lowerBounds = rep(0, d),
  upperBounds = rep(Inf, d), seed = 1, mean = m, precision = prec)

HZZtime <- sqrt(2) / sqrt(min(mgcv::slanczos(
  A = prec, k = 1,
  k1 = 1
)[['values']]))

currentSample <- rep(0.1, d)
for (i in 1:n) {
  m <- rnorm(d, 0, 1)
  prec <- rWishart(n = 1, df = d, Sigma = diag(d))[, , 1]
  setMean(sexp = engine$engine, mean = m)
  setPrecision(sexp = engine$engine, precision = prec)
  currentSample <- getZigzagSample(position = currentSample, nutsFlg = FALSE,
    engine = engine, stepZZHMC = HZZtime)
  samples[i,] <- currentSample
}

```

harmonicHMC	<i>Sample from a truncated Gaussian distribution with the harmonic HMC</i>
-------------	--

Description

Generate MCMC samples from a d-dimensional truncated Gaussian distribution with constraints $Fx+g \geq 0$ using the Harmonic Hamiltonian Monte Carlo sampler (Harmonic-HMC).

Usage

```
harmonicHMC(
  n,
  burnin = 0,
  mean,
  choleskyFactor,
  F,
  g,
  init,
  time = c(pi/8, pi/2),
  precFlg,
  diagnosticMode = FALSE
)
```

Arguments

n	number of samples after burn-in.
burnin	number of burn-in samples (default = 0).
mean	a d-dimensional mean vector.
choleskyFactor	upper triangular matrix R from Cholesky decomposition of precision or covariance matrix into $R^T R$.
F	F matrix (k-by-d matrix where k is the number of linear constraints).
g	g vector (k-dimensional).
init	a d-dimensional vector of the initial value. <code>init</code> must satisfy all constraints.
time	HMC integration time for each iteration. Can either be a scalar value for a fixed time across all samples, or a length 2 vector of a lower and upper bound for uniform distribution from which the time is drawn from for each iteration.
precFlg	logical. whether <code>choleskyFactor</code> is from precision (TRUE) or covariance matrix (FALSE).
diagnosticMode	logical. TRUE for also returning the bounce distances for each sample.

Value

List of samples: $(n + \text{burnin}) \times d$ matrix of samples (including burnin samples) and `bouncedDistances`: list of bounces for each sample (only present if `diagnosticMode` is TRUE).

References

Pakman A, Paninski L (2014). “Exact Hamiltonian Monte Carlo for truncated multivariate Gaussians.” *Journal of Computational and Graphical Statistics*, **23**(2), 518–542.

Examples

```
set.seed(1)
d <- 10
A <- matrix(runif(d^2)*2 - 1, ncol=d)
Sigma <- t(A) %**% A
R <- cholesky(Sigma)
mu <- rep(0, d)
F <- diag(d)
g <- rep(0,d)
initial <- rep(1, d)
results <- harmonicHMC(1000, 1000, mu, R, F, g, initial, precFlg = FALSE)
```

setMean	<i>Set the mean for the target MTN</i>
---------	--

Description

Set the mean vector for a given Zigzag-HMC engine object.

Usage

```
setMean(sexp, mean)
```

Arguments

sexp	pointer to a Zigzag-HMC engine object.
mean	the mean vector.

setPrecision	<i>Set the precision matrix for the target MTN</i>
--------------	--

Description

Set the precision matrix for a given Zigzag-HMC engine object.

Usage

```
setPrecision(sexp, precision)
```

Arguments

sexp	pointer to a Zigzag-HMC engine object.
precision	the precision matrix.

zigzagHMC

*Sample from a truncated Gaussian distribution***Description**

Generate MCMC samples from a d-dimensional truncated Gaussian distribution with element-wise truncations using the Zigzag Hamiltonian Monte Carlo sampler (Zigzag-HMC).

Usage

```
zigzagHMC(
  n,
  burnin = 0,
  mean,
  cov,
  prec = NULL,
  lowerBounds,
  upperBounds,
  init = NULL,
  step = NULL,
  nutsFlg = FALSE,
  rSeed = 1
)
```

Arguments

n	number of samples after burn-in.
burnin	number of burn-in samples (default = 0).
mean	a d-dimensional mean vector.
cov	a d-by-d covariance matrix of the Gaussian distribution. At least one of prec and cov should be provided.
prec	a d-by-d precision matrix of the Gaussian distribution.
lowerBounds	a d-dimensional vector specifying the lower bounds. $-\text{Inf}$ is accepted.
upperBounds	a d-dimensional vector specifying the upper bounds. Inf is accepted.
init	a d-dimensional vector of the initial value. <code>init</code> must satisfy all constraints. If <code>init = NULL</code> , a random initial value will be used.
step	step size for Zigzag-HMC or Zigzag-NUTS (if <code>nutsFlg = TRUE</code>). Default value is the empirically optimal choice: $\sqrt{2}(\lambda)^{-1/2}$ for Zigzag-HMC and $0.1(\lambda)^{-1/2}$ for Zigzag-NUTS, where λ is the minimal eigenvalue of the precision matrix.
nutsFlg	logical. If <code>TRUE</code> the No-U-Turn sampler will be used (Zigzag-NUTS).
rSeed	random seed (default = 1).

Value

an $(n + \text{burnin}) * d$ matrix of samples. The first burnin samples are from the user specified warm-up iterations.

References

Nishimura A, Zhang Z, Suchard MA (2021). “Hamiltonian zigzag sampler got more momentum than its Markovian counterpart: Equivalence of two zigzags under a momentum refreshment limit.” *arXiv preprint arXiv:2104.07694*.

Nishimura A, Dunson DB, Lu J (2020). “Discontinuous Hamiltonian Monte Carlo for discrete parameters and discontinuous likelihoods.” *Biometrika*, **107**(2), 365–380.

Examples

```
set.seed(1)
d <- 10
A <- matrix(runif(d^2)*2-1, ncol=d)
covMat <- t(A) %*% A
initial <- rep(1, d)
results <- zigzagHMC(n = 1000, burnin = 1000, mean = rep(0, d), cov = covMat,
lowerBounds = rep(0, d), upperBounds = rep(Inf, d))
```

Index

cholesky, [2](#)
createEngine, [2](#)
createNutsEngine, [3](#)

getInitialPosition, [4](#)
getZigzagSample, [4](#)

harmonicHMC, [6](#)

setMean, [7](#)
setPrecision, [7](#)

zigzagHMC, [8](#)