Package 'DEBBI'

July 21, 2025

Title Differential Evolution-Based Bayesian Inference

Version 0.1.0

Description Bayesian inference algorithms based on the population-based ``differential evolution" (DE) algorithm. Users can obtain posterior mode (MAP) estimates via DEMAP, posterior samples via DEMCMC, and variational approximations via DEVI.

License MIT + file LICENSE

URL https://github.com/bmgaldo/DEBBI

BugReports https://github.com/bmgaldo/DEBBI

Encoding UTF-8

Imports stats, doParallel, randtoolbox, numDeriv

Depends R (>= 3.5.0)

RoxygenNote 7.1.2

NeedsCompilation no

Author Brendan Matthew Galdo [aut, cre] (ORCID: <https://orcid.org/0000-0002-1279-3859>)

Maintainer Brendan Matthew Galdo <Brendan.m.galdo@gmail.com>

Repository CRAN

Date/Publication 2022-05-17 09:30:09 UTC

Contents

AlgoParamsDEMAP *AlgoParamsDEMAP*

Description

get control parameters for DEMAP function

Usage

```
AlgoParamsDEMAP(
  n_params,
  n_chains = NULL,
  n_iter = 1000,
  init_sd = 0.01,
  init_center = 0,
  n_cores_use = 1,
  step_size = NULL,
  jitter_size = 1e-06,
  crossover_rate = 1,
  parallel_type = "none",
  return_trace = FALSE,
  thin = 1
)
```

Arguments

n_params	number of free parameters estimated
n_chains	number of particle chains, 3*n_params is the default value
n_iter	number of iterations to run the sampling algorithm, 1000 is default
init_sd	positive scalar or n_params-dimensional numeric vector, determines the standard deviation of the Gaussian initialization distribution
init_center	scalar or n_params-dimensional numeric vector that determines the mean of the Gaussian initialization distribution
n_cores_use	number of cores used when using parallelization.
step_size	positive scalar, jump size in DE crossover step, default is 2.38/sqrt(2*n_params).
jitter_size	positive scalar, noise is added during crossover step from Uniform(-jitter_size,jitter_size) distribution. 1e-6 is the default value.
crossover_rate	number on the interval (0,1]. Determines the probability a parameter on a chain is updated on a given crossover step, sampled from a Bernoulli distribution.
parallel_type	string specifying parallelization type. 'none','FORK', or 'PSOCK' are valid values. 'none' is default value.
return_trace	logical, if true, function returns particle trajectories. This is helpful for diagnosing convergence or debugging model code. Function will return an iteration/thin x n_chains x n_params array and the estimated ELBO of each particle in a iteration/thin x n_chains array.

thin	positive integer, only every 'thin'-th iteration will be stored. Default value is 1. Increasing thin will reduce the memory required, while running chains for longer.

Value

list of control parameters for the DEMAP function

AlgoParamsDEMCMC *AlgoParamsDEMCMC*

Description

AlgoParamsDEMCMC

Usage

```
AlgoParamsDEMCMC(
  n_params,
  n_chains = NULL,
  param_names = NULL,
  n_iter = 1000,
  init_sd = 0.01,
  init_center = 0,
  n_cores_use = 1,
  step_size = NULL,
  jitter_size = 1e-06,
  parallel_type = "none",
  burnin = 0,
  thin = 1
)
```

Arguments

n_params	number of free parameters estimated
n_chains	number of MCMC chains, 3*n_params is the default value
param_names	optional vector of parameter names
n_iter	number of iterations to run the sampling algorithm, 1000 is default
init_sd	positive scalar or n_params-dimensional numeric vector, determines the standard deviation of the Gaussian initialization distribution
init_center	scalar or n_params-dimensional numeric vector, determines the mean of the Gaussian initialization distribution
n_cores_use	number of cores used when using parallelization.
step_size	positive scalar, jump size in DE crossover step, default is 2.38/sqrt(2*n_params) which is optimal for multivariate Gaussian target distribution (ter Braak, 2006)

jitter_size	positive scalar, noise is added during crossover step from Uniform(-jitter_size,jitter_size) distribution. 1e-6 is the default value.
parallel_type	string specifying parallelization type. 'none','FORK', or 'PSOCK' are valid values. 'none' is default value.
burnin	number of initial iterations to discard. Default value is 0.
thin	positive integer, only every 'thin'-th iteration will be stored. Default value is 1. Increasing thin will reduce the memory required, while running chains for longer.

Value

list of control parameters for the DEMCMC function

AlgoParamsDEVI *AlgoParamsDEVI*

Description

get control parameters for DEVI function

Usage

```
AlgoParamsDEVI(
  n_params,
  param_names = NULL,
  n_chains = NULL,
  n_iter = 1000,
  init_sd = 0.01,
  init_center = 0,
  n_cores_use = 1,
  step_size = NULL,
  jitter_size = 1e-06,
  parallel_type = "none",
  use_QMC = TRUE,
  purify = NULL,
  quasi_rand_seq = "halton",
  n_samples_ELBO = 10,
  LRVB_correction = TRUE,
  n_samples_LRVB = 25,
  neg_inf = -750,
  thin = 1,
  burnin = 0,
  return_trace = FALSE,
  crossover_rate = 1
)
```

Arguments

n_params	number of free parameters estimated
param_names	optional vector of parameter names
n_chains	number of particle chains used for optimization, 3*n_params is the default value
n_iter	number of iterations to run the sampling algorithm, 1000 is default
init_sd	positive scalar or n_params-dimensional numeric vector, determines the standard deviation of the Gaussian initialization distribution
init_center	scalar or n_params-dimensional numeric vector, determines the mean of the Gaussian initialization distribution
n_cores_use	number of cores used when using parallelization.
step_size	positive scalar, jump size in DE crossover step, default is 2.38/sqrt(2*n_params).
jitter_size	positive scalar, noise is added during crossover step from Uniform(-jitter_size,jitter_size) distribution. 1e-6 is the default value.
parallel_type	string specifying parallelization type. 'none','FORK', or 'PSOCK' are valid values. 'none' is default value.
use_QMC	logical, if true, a quasi-Monte Carlo estimator is used to estimate ELBO during optimization. default is TRUE.
purify	an integer, every 'purify'-th iteration, the Monte Carlo estimator of the ELBO is recalculated. This can help deal with noisy and outlier estimates of the ELBO. Default value is 25. If use_QMC is TRUE, purification is disabled as it is redundant.
quasi_rand_seq	type of low discrepancy sequence used for quasi Monte Carlo integration, either 'sobol' or 'halton'. LRVB correction always use QMC. Default is 'sobol'.
n_samples_ELBO	number of samples used for the Monte Carlo estimator of the ELBO (the objective function). default is 10.
LRVB_correction	logical, if true, LRVB covariance correction (Giordano, Brodderick, & Jordan 2018; Galdo, Bahg, & Turner 2020) is attempted.
n_samples_LRVB	number of samples used for LRVB correction. default is 25.
neg_inf	if density for a given value of theta is numerically 0 for q, this value is assigned for log density. This helps with numeric stability of algorithm. Default value is -750.
thin	positive integer, only every 'thin'-th iteration will be stored. Default value is 1. Increasing thin will reduce the memory required, while running algorithm for longer.
burnin	number of initial iterations to discard. Default value is 0.
return_trace	logical, if true, function returns particle trajectories. This is helpful for diagnosing convergence or debugging model code. Function will return an iteration/thin x n_chains x n_params array and the estimated ELBO of each particle in a iteration/thin x n_chains array.
crossover_rate	number on the interval (0,1]. Determines the probability a parameter on a chain is updated on a given crossover step, sampled from a Bernoulli distribution.

Value

list of control parameters for the DEVI function

DEMAP *DEMAP*

Description

DE optimization for maximum a posteriori (MAP) estimation; his function tries to find the posterior mode.

Usage

```
DEMAP(LogPostLike, control_params = AlgoParamsDEMAP(), ...)
```

Arguments

LogPostLike	function whose first argument is an n_params-dimensional model parameter vector and returns (scalar) sum of log prior density and log likelihood for the parameter vector.
control_params	control parameters for DE algorithm. see AlgoParamsDEMAP function documentation for more details.
...	additional arguments to pass LogPostLike

Value

list contain posterior samples from DEMCMC in a n_iters_per_chain by n_chains by n_params array and the log likelihood of each sample in a n_iters_per_chain by n_chains array.

Examples

```
# simulate from model
dataExample <- matrix(stats::rnorm(100, c(-1, 1), c(1, 1)), nrow = 50, ncol = 2, byrow = TRUE)

# list parameter names
param_names_example <- c("mu_1", "mu_2")

# log posterior likelihood function = log likelihood + log prior | returns a scalar
LogPostLikeExample <- function(x, data, param_names) {
  out <- 0

  names(x) <- param_names

  # log prior
  out <- out + sum(dnorm(x["mu_1"], 0, sd = 1, log = TRUE))
  out <- out + sum(dnorm(x["mu_2"], 0, sd = 1, log = TRUE))

  # log likelihoods
```

```
  out <- out + sum(dnorm(data[, 1], x["mu_1"], sd = 1, log = TRUE))
  out <- out + sum(dnorm(data[, 2], x["mu_2"], sd = 1, log = TRUE))

  return(out)
}

# Get map estimates
DEMAP(
  LogPostLike = LogPostLikeExample,
  control_params = AlgoParamsDEMAP(
    n_params = length(param_names_example),
    n_iter = 1000,
    n_chains = 12
  ),
  data = dataExample,
  param_names = param_names_example
)
```

DEMCMC *DEMCMC*

Description

Sample from posterior using Differential Evolution Markov Chain Monte Carlo

Usage

```
DEMCMC(LogPostLike, control_params = AlgoParamsDEMCMC(), ...)
```

Arguments

LogPostLike	function whose first argument is an n_params-dimensional model parameter vector and returns (scalar) sum of log prior density and log likelihood for the parameter vector.
control_params	control parameters for DEMCMC algorithm. see AlgoParamsDEMCMC function documentation for more details. You must specify 'n_params' here.
...	additional arguments to pass LogPostLike

Value

list contain posterior samples from DEMCMC in a 'n_samples_per_chain' by 'n_chains' by n_params array and the log posterior likelihood of each sample in a 'n_samples_per_chain' by 'n_chains' array.

Examples

```
# simulate from model
dataExample <- matrix(stats::rnorm(100, c(-1, 1), c(1, 1)), nrow = 50, ncol = 2, byrow = TRUE)
#
# list parameter names
param_names_example <- c("mu_1", "mu_2")

# log posterior likelihood function = log likelihood + log prior | returns a scalar
LogPostLikeExample <- function(x, data, param_names) {
  out <- 0

  names(x) <- param_names

  # log prior
  out <- out + sum(dnorm(x["mu_1"], 0, sd = 1, log = TRUE))
  out <- out + sum(dnorm(x["mu_2"], 0, sd = 1, log = TRUE))

  # log likelihoods
  out <- out + sum(dnorm(data[, 1], x["mu_1"], sd = 1, log = TRUE))
  out <- out + sum(dnorm(data[, 2], x["mu_2"], sd = 1, log = TRUE))

  return(out)
}

# Sample from posterior
DEMCMC(
  LogPostLike = LogPostLikeExample,
  control_params = AlgoParamsDEMCMC(
    n_params = length(param_names_example),
    n_iter = 1000,
    n_chains = 12
  ),
  data = dataExample,
  param_names = param_names_example
)
```

DEVI *DEVI*

Description

DE optimization for mean-field variational inference. Minimizes the KL divergence (maximizes the ELBO) between $q(thetallambda)$ and the target posterior $p(thetaldata)$ For a tutorial on variational inference check out Galdo, Bahg, & Turner 2020.

Usage

```
DEVI(LogPostLike, control_params = AlgoParamsDEVI(), ...)
```

Arguments

LogPostLike	function whose first argument is an n_params-dimensional model parameter vector and returns (scalar) sum of log prior density and log likelihood for the parameter vector.
control_params	control parameters for DE algorithm. see AlgoParamsDEVI function documentation for more details.
...	additional arguments to pass LogPostLike

Value

list contain mean in a n_iters_per_chain by n_chains by 2*n_params_model array and the ELBO of each sample in a n_iters_per_chain by n_chains array.

Examples

```
# simulate from model
dataExample <- matrix(stats::rnorm(100, c(-1, 1), c(1, 1)), nrow = 50, ncol = 2, byrow = TRUE)
## list parameter names
param_names_example <- c("mu_1", "mu_2")

# log posterior likelihood function = log likelihood + log prior | returns a scalar
LogPostLikeExample <- function(x, data, param_names) {
  out <- 0

  names(x) <- param_names

  # log prior
  out <- out + sum(dnorm(x["mu_1"], 0, sd = 1, log = TRUE))
  out <- out + sum(dnorm(x["mu_2"], 0, sd = 1, log = TRUE))

  # log likelihoods
  out <- out + sum(dnorm(data[, 1], x["mu_1"], sd = 1, log = TRUE))
  out <- out + sum(dnorm(data[, 2], x["mu_2"], sd = 1, log = TRUE))

  return(out)
}

# Get variational approximation
DEVI(
  LogPostLike = LogPostLikeExample,
  control_params = AlgoParamsDEVI(
    n_params = length(param_names_example),
    n_iter = 200,
    n_chains = 12
  ),
  data = dataExample,
  param_names = param_names_example
)
```

Index