Package ‘iterLap’

October 1, 2023

Type Package
Title Approximate Probability Densities by Iterated Laplace Approximations
Version 1.1-4
Depends quadprog, randtoolbox, parallel, R (>= 2.15)
Date 2023-09-29
Author Bjoern Bornkamp
Maintainer Bjoern Bornkamp <bbnkmp@mail.de>
Description The iterLap (iterated Laplace approximation) algorithm approximates a general (possibly non-normalized) probability density on \( \mathbb{R}^p \), by repeated Laplace approximations to the difference between current approximation and true density (on log scale). The final approximation is a mixture of multivariate normal distributions and might be used for example as a proposal distribution for importance sampling (e.g. in Bayesian applications).

The algorithm can be seen as a computational generalization of the Laplace approximation suitable for skew or multimodal densities.

License GPL
LazyLoad yes
NeedsCompilation yes
Repository CRAN
Date/Publication 2023-09-30 22:30:02 UTC

R topics documented:

iterLap-package .................................................. 2
GRApprox .......................................................... 3
Importance Sampling and independence Metropolis Hastings sampling ....... 4
iterLap .............................................................. 5
resample ............................................................ 7

Index 9
iterLap-package

iterLap package information

Description

Implementation of iterLap

Details

This package implements the multiple mode Laplace approximation by Gelman and Rubin (via function GRApprox) and the iterated Laplace approximation (via the function iterLap). Both functions return objects of class mixDist, which contain the fitted mode vectors and covariance matrices. Print and summary methods exist to display the contents of a mixDist object in human-readable form. Function IS performs importance sampling, using a mixDist object as input parameter.

Author(s)

Bjoern Bornkamp
Maintainer: Bjoern Bornkamp <bbnkmp@gmail.com>

References


Examples

```r
## banana example
banana <- function(pars, b, sigma12){
dim <- 10
cc <- c(1/sqrt(sigma12), rep(1, dim-1))
return(-0.5*sum((y*cc)^2))
}

start <- rbind(rep(0,10),rep(-1.5,10),rep(1.5,10))
## multiple mode Laplace approximation
gr <- GRApprox(banana, start, b = 0.03, sigma12 = 100)
## print mixDist object
gr
## summary method
summary(gr)
## importance sampling using the obtained mixDist object
## using a mixture of t distributions with 10 degrees of freedom
issamp <- IS(gr, nSim=1000, df = 10, post=banana, b = 0.03, sigma12 = 100)
## effective sample size
issamp$ESS
```
## now use iterated Laplace approximation (using gr mixDist object from above as starting approximation)
iL <- iterLap(banana, GRobj = gr, b = 0.03, sigma2 = 100)
IS Obj <- IS(iL, nSim=10000, df = 100, post=banana, b = 0.03, sigma2 = 100)
## residual resampling to obtain unweighted sample
sims <- resample(1000, IS Obj)
plot(sims[,1], sims[,2], xlim=c(-40,40), ylim = c(-40,20))

---

**GRApprox**  

### Gelman-Rubin mode approximation

**Description**

Performs the multiple mode approximation of Gelman-Rubin (applies a Laplace approximation to each mode). The weights are determined corresponding to the height of each mode.

**Usage**

```r
GRApprox(post, start, grad, method = c("nlminb", "nlm", "Nelder-Mead", "BFGS"),
          control = list(), ...)
```

**Arguments**

- `post` log-posterior density.
- `start` vector of starting values if dimension=1 otherwise matrix of starting values with the starting values in the rows
- `grad` gradient of log-posterior
- `method` Which optimizer to use
- `control` Control list for the chosen optimizer
- `...` Additional arguments for log-posterior density specified in `post`

**Value**

Produces an object of class mixDist. That a list mit entries `weights` Vector of weights for individual components, `means` Matrix of component medians of components, `sigmas` List containing scaling matrices, `eigenHess` List containing eigen decompositions of scaling matrices, `dets` Vector of determinants of scaling matrix, `sigmainv` List containing inverse scaling matrices

**Author(s)**

Bjoern Bornkamp
References


See Also

iterLap

Examples

```r
## log-density for banana example
banana <- function(pars, b, sigma12){
  dim <- 10
  cc <- c(1/sqrt(sigma12), rep(1, dim-1))
  return(-0.5*sum((y*cc)^2))
}

start <- rbind(rep(0,10),rep(-1.5,10),rep(1.5,10))
## multiple mode Laplace approximation
aa <- GRApprox(banana, start, b = 0.03, sigma12 = 100)
## print mixDist object
aa
## summary method
summary(aa)
## importance sampling using the obtained mixDist object
## using a mixture of t distributions with 10 degrees of freedom
dd <- IS(aa, nSim=1000, df = 10, post=banana, b = 0.03, sigma12 = 100)
## effective sample size
dd$ESS
```

Description

Monte Carlo sampling using the iterated Laplace approximation.

Usage

- `IS(obj, nSim, df = 4, post, vectorized = FALSE, cores = 1, ...)`
- `IMH(obj, nSim, df = 4, post, vectorized = FALSE, cores = 1, ...)`
### iterLap

**Iterated Laplace Approximation**

Arguments

- **obj**: an object of class "mixDist"
- **nSim**: number of simulations
- **df**: degrees of freedom of the mixture of t distributions proposal
- **post**: log-posterior density
- **vectorized**: Logical determining, whether post is vectorized
- **cores**: number of cores you want to use to evaluate the target density (uses the mclapply function from the parallel package). For Windows machines, a value > 1 will have no effect, see mclapply help for details.
- **...**: additional arguments passed to `post`.

Value

A list with entries:
- **samp**: Matrix containing sampled values
- **w**: Vector of weights for values in samp
- **normconst**: normalization constant estimated based on importance sampling
- **ESS**: Effective sample size (for IS)
- **accept**: Acceptance rate (for IMH)

Author(s)

Bjoern Bornkamp

Examples

```r
## see function iterLap for an example on how to use IS and IMH
```

```r
iterLap(post, ..., GRobj = NULL, vectorized = FALSE, startVals = NULL, method = c("nlminb", "nlm", "Nelder-Mead", "BFGS"), control = NULL, nlcontrol = list())
```
Arguments

- `post` log-posterior density
- ... additional arguments to log-posterior density
- `GRobj` object of class `mixDist`, for example resulting from a call to `GRApprox`
- `vectorized` Logical determining, whether `post` is vectorized
- `startVals` Starting values for `GRApprox`, when `GRobj` is not specified. Vector of starting values if `dimension=1` otherwise matrix of starting values with the starting values in the rows
- `method` Type of optimizer to be used.
- `control` List with entries:
  - `gridSize` Determines the size of the grid for each component
  - `delta` Stopping criterion based on the maximum error on the grid
  - `maxDim` Maximum number of components allowed (default 20)
  - `eps` Stopping criterion based on normalization constant of approximation
  - `info` How much information should be displayed during iterations: 0 - none, 1 - minimum information, 2 - maximum information
- `nlcontrol` Control list for the used optimizer.

Value

Produces an object of class `mixDist`: A list with entries
- `weights` Vector of weights for individual components
- `means` Matrix of component medians of components
- `sigmas` List containing scaling matrices
- `eigenHess` List containing eigen decompositions of scaling matrices
- `dets` Vector of determinants of scaling matrix
- `sigmainv` List containing inverse scaling matrices

Author(s)

Bjoern Bornkamp

References


Examples

```r
#### banana example
banana <- function(pars, b, sigma12){
  dim <- 10
  cc <- c(1/sqrt(sigma12), rep(1, dim-1))
  return(-0.5*sum((y*cc)^2))
}```
## resample

### Residual resampling

#### Description

Perform residual resampling to the result of importance sampling

---

```r
resample = 7
```
Usage
resample(n, obj)

Arguments
n Number of resamples to draw
obj An object of class IS, as produced by the IS function

Value
Matrix with resampled values

Author(s)
Bjoern Bornkamp

Examples
### see function iterLap for an example on how to use resample
Index

* misc
  GRAprox, 3
  Importance Sampling and independence Metropolis Hastings sampling, 4
  iterLap, 5
  resample, 7

* package
  iterLap-package, 2

GRAprox, 3

IMH (Importance Sampling and independence Metropolis Hastings sampling), 4
Importance Sampling and independence Metropolis Hastings sampling, 4
IS (Importance Sampling and independence Metropolis Hastings sampling), 4
iterLap, 4, 5
iterLap-package, 2

resample, 7