Package 'optimParallel'

October 14, 2022

Type Package

Title Parallel Version of the L-BFGS-B Optimization Method

Version 1.0-2

Date 2021-02-10

Maintainer Florian Gerber <flora.fauna.gerber@gmail.com>

Description Provides a parallel version of the L-BFGS-B method of optim(). The main function of the package is optimParallel(), which has the same usage and output as optim(). Using optimParallel() can significantly reduce the optimization time.

License GPL (>= 2)

URL https://github.com/florafauna/optimParallel-R

BugReports https://github.com/florafauna/optimParallel-R/issues

Depends R (>= 3.5), stats, parallel

Suggests R.rsp, roxygen2, spam, microbenchmark, testthat, ggplot2, numDeriv, lbfgsb3c

VignetteBuilder R.rsp

RoxygenNote 7.1.1

NeedsCompilation no

Author Florian Gerber [aut, cre] (<https://orcid.org/0000-0001-8545-5263>)

Repository CRAN

Date/Publication 2021-02-11 16:00:06 UTC

R topics documented:

optimParallel 2

Index **6**

optimParallel *parallel version of the L-BFGS-B method of* optim

Description

The function provides a parallel version of the L-BFGS-B method of optim. If the evaluation time of the objective function fn is more than 0.1 sceconds, optimParallel can significantly reduce the optimization time. For a p-parameter optimization the speed increase is about factor $1 + 2p$ when no analytic gradient is specified and $1 + 2p$ processor cores are available.

Usage

```
optimParallel(
  par,
  fn,
  gr = NULL,
  ...,
  lower = -Inf,
  upper = Inf,
  control = list(),
  hessian = FALSE,
  parallel = list()
)
```

Arguments

par	see the documentation of optim.
fn	see the documentation of optim.
gr	see the documentation of optim.
...	see the documentation of optim. See section 'Notes' for more information.
lower	see the documentation of optim.
upper	see the documentation of optim.
control	see the documentation of optim.
hessian	see the documentation of optim.
parallel	is a list of additional control parameters and can supply any of the following components:

cl an object of class "cluster" specifying the cluster to be used for parallel execution. See makeCluster for more information. If the argument is not specified or NULL, the default cluster is used. See setDefaultCluster for information on how to set up a default cluster.

forward logical vector of length 1. If FALSE (default when loading the package), a numeric central difference approximation of the gradient defined as $(fn(x + \epsilon) - fn(x - \epsilon))/(2\epsilon)$ is used, which corresponds to the gradient

approximation used in optim. If TRUE, a numeric forward difference approximation of the gradient essentially defined as $(fn(x+\epsilon) - fn(x))/\epsilon$ is used. This reduces the number of function calls from $1 + 2p$ to $1 + p$ and can be useful if the number of available cores is smaller than $1 + 2p$ or if the memory limit is reached. Note that the numeric central difference approximation is more accurate than the numeric forward difference approximation.

`loginfo` logical vector of length 1 with default value FALSE when loading the package. If TRUE, additional log information containing the evaluated parameters as well as return values of fn and gr is returned.

Details

optimParallel is a wrapper to optim and relies on the lexical scoping mechanism of R and the R package **parallel** to evaluate fn and its (approximate) gradient in parallel.

Some default values of the argument parallel can be set via
`options("optimParallel.forward", "optimParallel.loginfo")`.

Value

Same as the return value of optim. See the documentation thereof for more information.
If `parallel=list(loginfo=TRUE)`, additional log information containing the evaluated parameters as well as the return values of fn and gr is returned.

Notes

1. If fn or gr depend on functions or methods from loaded packages, it may be necessary to explicitly load those packages in all processes of the cluster. For cl of class "cluster" one can use `clusterEvalQ(cl, search())` to check whether all required packages are on the search paths of all processes. If, for example, the R package **spam** is required and missing on those search paths, it can be added via `clusterEvalQ(cl, library("spam"))`.
2. If fn or gr have more than one argument, it may be necessary to pass those to optimParallel via the ... argument. An illustration is given in the section 'Examples'.
3. We recommend that all R objects used by fn and/or gr are passed to fn and/or gr via arguments. In certain cases it may also work that fn and/or gr use objects from the .GlobalEnv (without having corresponding arguments). In that case it can be necessary to pass those objects to all processes of the used cluster via clusterExport. An illustration is given in the section 'Examples'.
4. Using parallel R code inside fn and gr can work if suitable clusters are setup (one cluster for optimParallel and one for the parallel execution of fn and gr).
5. Using optimParallel with n parallel processes increases the memory usage by about factor n compared to a call to optim. If the memory limit is reached this may severely slowdown the optimization. Strategies to reduce memory usage are (1) kill all unused processes on the computer, (2) revise the code of fn and/or gr to reduce its memory usage, and (3) reduce the number of parallel processes by specifying the argument `parallel=list(forward=TRUE)` and/or setting up a cluster with less parallel processes.

Issues and bug report

A list of known issues of optimParallel can be found at https://github.com/florafauna/optimParallel-R/issues. Please report issues not listed there to <flora.fauna.gerber@gmail.com>. Do not forget to include an R script reproducing the issue and the output of sessionInfo().

Author(s)

Florian Gerber, <flora.fauna.gerber@gmail.com>, https://user.math.uzh.ch/gerber.

References

F. Gerber, R. Furrer (2019) optimParallel: An R package providing a parallel version of the L-BFGS-B optimization method. The R Journal, 11(1):352-358, https://doi.org/10.32614/RJ-2019-030 Also available as vignette of this package vignette("optimParallel").

See Also

optim, makeCluster, setDefaultCluster, stopCluster, detectCores.

Examples

```
negll <- function(par, x, sleep=0, verbose=TRUE){
    if(verbose)
        cat(par, "\n")
   Sys.sleep(sleep)
   -sum(dnorm(x=x, mean=par[1], sd=par[2], log=TRUE))
}
set.seed(13); x <- rnorm(1000, 5, 2)

cl <- makeCluster(2)     # set the number of processor cores
setDefaultCluster(cl=cl) # set 'cl' as default cluster

optimParallel(par=c(1,1), fn=negll, x=x, lower=c(-Inf, .0001))

optimParallel(par=c(1,1), fn=negll, x=x, sleep=0, verbose=TRUE,
              lower=c(-Inf, .0001), parallel=list(loginfo=TRUE))

setDefaultCluster(cl=NULL); stopCluster(cl)

## default values of the argument 'parallel':
options("optimParallel.forward", "optimParallel.loginfo")

## Not run:
## - use all avilable processor cores
## - return cat() output to R prompt
##   (may have issues on Windows)
if(tolower(.Platform$OS.type) != "windows"){
    cl <- makeCluster(spec=detectCores(), type="FORK", outfile="")
} else
    cl <- makeCluster(spec=detectCores(), outfile="")
setDefaultCluster(cl=cl)
```

```
## return log information
options(optimParallel.loginfo=TRUE)

## stop if change of f(x) is smaller than 0.01
control <- list(factr=.01/.Machine$double.eps)

optimParallel(par=c(1,1), fn=negll, x=x, sleep=.5, verbose=TRUE,
              verbose=TRUE, lower=c(-Inf, .0001), control=control)
## each step invokes 5 parallel calls to negll()

optimParallel(par=c(1,1), fn=negll, x=x, sleep=.5, verbose=TRUE,
              lower=c(-Inf, .0001), control=control,
              parallel=list(forward=TRUE))
## each step invokes 3 parallel calls to negll()

## passing objects to fn/gr (see section 'Notes')
## ----------------------------------------------
a <- 10
fn <- function(par, b) sum((par-a-b)^2)

## approach 1:
clusterExport(cl, "a")
optimParallel(par=1, fn=fn, b=1)

## approach 2 (recommended):
## rewrite 'fn' such that all necessary objects
## are passed as arguments
fn <- function(par, a, b) sum((par-a-b)^2)
optimParallel(par=1, fn=fn, a=20, b=1)

setDefaultCluster(cl=NULL); stopCluster(cl)
## End(Not run)
```

Index

∗ **package**
optimParallel, 2

clusterExport, *3*

detectCores, *4*

makeCluster, *2*, *4*

optim, *2–4*
optimParallel, 2
optimparallel *(*optimParallel*)*, 2
OptimParallel-Package *(*optimParallel*)*, 2
OptimParallel-package *(*optimParallel*)*, 2
optimParallel-Package *(*optimParallel*)*, 2
optimParallel-package *(*optimParallel*)*, 2
optimparallel-Package *(*optimParallel*)*, 2
optimparallel-package *(*optimParallel*)*, 2

setDefaultCluster, *2*, *4*
stopCluster, *4*